(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 521 729 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
15.12.1999 Bulletin 1999/50

(51) Int Cl.6: **G06F 15/80**

(21) Application number: **92306160.0**

(22) Date of filing: **03.07.1992**

(54) Neural network and learning method therefor

Neuronales Netzwerk und Lernverfahren

Réseau neuronal et méthode d'apprentissage

(84) Designated Contracting States:
DE FR GB

(30) Priority: **03.07.1991 JP 16314291**

(43) Date of publication of application:
07.01.1993 Bulletin 1993/01

(73) Proprietor: **HONDA GIKEN KOGYO KABUSHIKI KAISHA**
Minato-ku Tokyo 107 (JP)

(72) Inventor: **Watanabe, Ryujin**
1-chome, Wako-shi, Saitama-ken (JP)

(74) Representative: **Tothill, John Paul et al**
Frank B. Dehn & Co.
179 Queen Victoria Street
London EC4V 4EL (GB)

(56) References cited:
EP-A- 0 360 674

- **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, vol.36, no.5, May 1989, NEW YORK US pages 661 - 664 MIRCHANDANI 'On hidden nodes for neural networks'**
- **INNC90 INTERNATIONAL NEURAL NETWORK CONFERENCE, vol.2, 9 July 1990, PARIS , FRANCE pages 877 - 880 ARAI 'Conditions on activation functions of hidden units for learning by backpropagation'**
- **IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, vol.1, 24 July 1988, SAN DIEGO , USA pages 247 - 254 KOUTSOUGERAS 'Training of a neural network for pattern classification based on an entropy measure'**
- **IJCNN INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, vol.3, 17 June 1990, SAN DIEGO , USA pages 245 - 250 KONG 'The piecewise linear neural network: training and recognition'**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 521 729 B1

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] The present invention relates to a neural network, and more particularly to a neural network that can be trained to learn a linearly inseparable input pattern and output values therefor (the combination of such a linearly inseparable input pattern and output values therefor will be referred to as an "input/output pattern"), and to a learning method for said neural network.

[0002] Various models for neural networks have heretofore been proposed. It is known that the perceptron is the most basic neural network model. The perceptron is based on the learning process in which only the weight of association (association coefficient) between input and output layers is varied. The learning process of the perceptron is advantageous in that it is simpler than other learning processes. The learning process is a process in which the weight is varied to get the output to approach a target output.

[0003] However, the perceptron is has a problem in that it cannot be trained to learn linearly inseparable patterns. If a pattern that can be classified into two kinds is linearly separable, it can be separated into the classifications in a straight line when it is two-dimensional and in a multidimensional plane when it is three-dimensional or more.

[0004] The problem of the perceptron in that linearly inseparable patterns cannot be learned can be solved by a process known as back propagation. The back propagation is applicable to a multilayer neural network model, and allows linearly inseparable patterns to be learned. The back propagation basically serves to minimise the square function of an error between a target output and an actual output, and uses an optimization process referred to as the method of steepest descent. Therefore, if there is a local minimum present in an error function obtained from given input/output patterns, then a learning failure may result when trapped in the local minimum. The back propagation is also disadvantageous in that it is cumbersome to adjust parameters and initialize the weight, and difficult to determine the number of necessary neurons of a hidden layer between input and output layers, and the process requires a large amount of calculation and is time-consuming.

[0005] There have also been proposed algorithms for adding neuron units needed to construct a multilayer neural network.

[0006] One of the proposed algorithms serves to determine the number of neuron unit layers of a feedforward hierarchical network and the number of neuron units in each of the neuron unit layers. It adds neuron unit layers or neuron units in neuron unit layers until convergence is reached. For details, see *The Upstart Algorithm: A method for Constructing and Training Feedforward Neural Networks,* written by Marcus Frean, Neural Computation, Vol. 2, pages 198 - 209, 1990, Massachusetts Institute of Technology.

[0007] According to another proposed algorithm, neuron units are added as required according to predetermined rules in order to build a multilayer perceptron composed of linear threshold units. For details, see *Learning in feedforward layered networks: the tiling algorithm,* written by Marc Mézard and Jean-Pierre Nadal, Journal of Physics A: Math. Gen. Vol. 22, pages 2191 - 2203, 1989.

[0008] The above proposed algorithms add necessary neuron units until a desired output is obtained with respect to a given input. Therefore, neuron layers and neurons thereof which are required are not determined until a neural network is finally constructed, and the number of neuron layers used and the number of neurons used tend to be large. As a result, the resultant neuron networks are liable to be complex in structure, and do not lend themselves to high-speed processing operation.

[0009] It is an object of the present invention to provide a neural network which solves the problem of linear inseparability of the perceptron and the problem of being trapped in a local minimum of back propagation, and which can learn a linearly inseparable pattern with as few neurons as possible, and also a learning method for automatically generating hidden neurons necessary for such a neural network.

[0010] According to the present invention, a neural network is provided as set out in claim 1.

[0011] According to the present invention, a learning method for a neural network is also provided as set out in claim 5.

[0012] The hidden neurons of the hidden layer between the input and output layers are automatically generated, so that a linearly inseparable input/output pattern is divided into linearly separable input/output patterns, which are then combined together. As a whole, the neural network is capable of learning input/output patterns that are linearly inseparable. Since the input and output layers are provided in advance, and only a minimum number of hidden neurons or a number of hidden neurons close thereto are determined, the neural network is highly versatile, and can operate at high speed for information processing.

[0013] Experimental results, described later on, indicate that the learning method was able to learn all input/output patterns with four inputs and one output. The learning method had a learning speed about 500 times faster than the learning speed of the back propagation process. Patterns whose inputs are all 0 can be learned by expanding or modifying the basic arrangement of the neural network and the learning method according to the present invention.

[0014] Some embodiments of the invention will now be described by way of example and with reference to the accompanying drawings, in which

FIG. 1 is a block diagram illustrating the fundamental concept of the present invention;
FIGS. 2(a) through 2(f) are diagrams showing a principle for transforming a linearly unseparable pattern into a linearly separable pattern;
FIG. 3 is a diagram of a neural network according to an embodiment of the present invention;
FIG. 4 is a diagram showing a linearly separable allocation process;
FIGS. 5(a) and 5(b) are diagrams showing the positional relationship between two pattern sets in a state variable space of input neurons;
FIGS. 6(a) and 6(b) are diagrams illustrative of transformation of a linearly unseparable pattern;
FIGS. 7(a) and 7(b) are diagrams showing transformation functions for two kinds of patterns;
FIG. 8 is a flowchart of a learning method according to an embodiment of the present invention;
FIG. 9 is a flowchart of a transformation routine in the flowchart of FIG. 8;
FIGS. 10(a) through 10(c) are diagrams showing a pattern transformation for speeding up algorithms;
FIGS. 11 and 12 are diagrams showing all algorithms in detail;
FIG. 13 is a diagram of definitions of variables shown in FIGS. 11 and 12;
FIGS. 14 through 16 are a flowchart of an algorithm γ used in the learning method;
FIG. 17 is a flowchart of an algorithm δ^k shown in FIG. 16;
FIG. 18 is a diagram of a neural network with bias neurons;
FIG. 19 is a diagram showing a bias neuron unit with a self feedback loop;
FIG. 20 is a diagram of a neural network with multiple inputs and multiple outputs;
FIG. 21 is a diagram of a multilayer neural network having a plurality of hidden layers;
FIG. 22 is a logarithmic graph of calculating times of the learning method of the present invention and the back propagation method; and
FIG. 23 is a graph showing percentages of correct answers according to the learning method of the present invention and the back propagation method with respect to the number of patterns.

[0015] FIG. 1 shows the fundamental concept of the present invention. According to the present invention, as shown in FIG. 1, a linearly inseparable input/output pattern P is divided into linearly separable patterns $Q_1, Q_2, \cdots, Q_n$, and a combination pattern R for combining the linearly separable patterns $Q_1, Q_2, \cdots, Q_n$ is generated. The linearly separable patterns $Q_1, Q_2, \cdots, Q_n$ are implemented by hidden neurons, and the combination pattern R is implemented by output neurons, so that the linearly inseparable input/output pattern P can be learned as a whole.

[0016] According to the present invention, a neural network having input, output, and hidden layers comprises a lower neural network model composed of hidden layer neurons and input layer neurons for learning the linearly separable patterns $Q_1, Q_2, \cdots, Q_n$, and a higher neural network model composed of hidden layer neurons and output layer neurons for combining the linearly separable patterns $Q_1, Q_2, \cdots, Q_n$ into the combination pattern R.

[0017] The learning method according to the present invention effects a predetermined learning process on a feedforward neural network for generating a plurality of linearly separable patterns from a given linearly inseparable pattern.

[0018] The neural network and the learning method therefor will hereinafter be described in greater detail.

(1) Formation of a neural network:

[0019] Any optional input/output pattern including a linearly inseparable pattern P can be represented by a combination of linearly separable patterns $Q_1, Q_2, \cdots, Q_n$.

[0020] For example, as shown in FIG. 2(a), two patterns indicated respectively by black dots (ON) and white dots (OFF) are presented in a multidimensional conceptual coordinate system that is conceptually represented by two-dimensional coordinates. The issue here is whether these patterns are linearly separable.

[0021] Now, straight lines a, b, c, d as shown in FIG. 2(b) are introduced into the patterns. The patterns shown in FIG. 2(a) can be expressed by sum and difference sets in regions indicated by respective upwardly directed arrows on the straight lines.

[0022] The characters typed in bold represent vectors in the following description.

[0023] The straight lines a, b, c, d are represented as follows:

$$a : \mathbf{w}_A^T \mathbf{x} = \theta,$$

$$b : \mathbf{w}_B^T \mathbf{x} = \theta,$$

$$c : w_C{}^T x = \theta,$$

$$d : w_D{}^T x = \theta$$

where **w** is a association weight vector, **x** is a state value vector of input neurons, ^{T} represents transposition, and θ is a threshold. In FIG. 2(b), $w_A > 0, w_B > 0$, $w_C > 0$, $w_D > 0$, and $\theta \geq 0$.

[0024] As shown in FIGS. 2(c) through 2(f), the sets of the black dots in the regions indicated by the upwardly directed arrows on the straight lines are contained in the following regions:

$$\text{Set of black dots in FIG. 2 (c)} : w_A{}^T x > \theta;$$

$$\text{Set of black dots in FIG. 2 (d)}: w_B{}^T x > \theta;$$

$$\text{Set of black dots in FIG. 2 (e)}: w_C{}^T x > \theta;$$

and

$$\text{Set of black dots in FIG. 2 (f)} : w_D{}^T x > \theta.$$

[0025] These regions or sets are indicated by A, B, C, D, respectively. If an operation to determine a sum set is indicated by a symbol <+> and an operation to determine a difference set is indicated by a symbol <->, then the two types of patterns that are separated as shown in FIG. 2(b) are expressed as follows:

$$\{(A <+> B) <+> C\} <-> D.$$

[0026] Whether the variable vector **x** in the regions A, B, C, D is contained in the regions or not can be indicated by whether the values of $L(w_A{}^T x - \theta)$, $L(w_B{}^T x - \theta)$, $L(w_C{}^T x - \theta)$, $L(w_D{}^T x - \theta)$ are 1 or 0, where L is a threshold function that is 1 or 0 depending on a variable z as follows:

when $z \geq 0$, $L(z) = 1$, and
when $z < 0$, $L(z) = 0$.

[0027] If any of the above values is 1, then the variable vector **x** is present in the region indicated by **w.**

[0028] If it is assumed that the above values are expressed as follows:

$$x_A = L(w_A{}^T x - \theta),$$

$$x_B = L(w_B{}^T x - \theta),$$

$$x_C = L(w_C{}^T x - \theta),$$

and

$$x_D = L(w_D{}^T x - \theta),$$

then the patterns shown in FIG. 2(b) can be expressed by the following equation:

$$y = L((\theta + \varepsilon)x_A - 2\varepsilon x_B + 2\varepsilon x_C - 2\varepsilon x_D - \theta)$$

where ε is a positive number and $\varepsilon < \theta$.

[0029] The coefficients, indicated by θ and ε, of x_A, x_B, x_C, x_D may not strictly be of the values given by the above equations, but may be of any values or, broadly stated, may not be defined by the above equations, insofar as the input/output relationship between x_A, x_B, x_C, X_D and y is the same as the above equations.

[0030] y has a value of 1 with respect to the black dots in FIG. 2(a) and a value of 0 with respect to the white dots in FIG. 2(a). In this manner, even if the original input/output pattern is linearly inseparable, it can be transformed into linearly separable patterns, and can be expressed by neural network models.

[0031] The above process generalized by replacing x_A, x_B, x_C, x_D with x^1, x^2, x^3, x^4, respectively, and rewriting the above equations as follows:

$$x^1 = L(w_A{}^T x - \theta),$$

$$x^2 = L(w_B{}^T x - \theta),$$

$$x^3 = L(w_C{}^T x - \theta),$$

$$x^4 = L(w_D{}^T x - \theta),$$

and

$$y = L((\theta + \varepsilon)x^1 - 2\varepsilon x^2 + 2\,\varepsilon x^3 - 2\varepsilon x^4 - \theta)$$

If necessary, x^5, x^6, x^7, ··· may be added.

[0032] The above general equations apply to a three-layer neural network having an input layer 1, a hidden layer 2, and an output layer 3 as shown in FIG. 3, and indicate the relationship between state values x^1, x^2, x^3, ··· of neurons of the hidden layer 2 and the state value y of neurons of the output layer 3. The input layer 1 is composed of a plurality of n neurons whose state values are indicated by $x_1, x_2, \cdots, x_n$, and the output layer 3 is composed of a single neuron whose state value is indicated by y. These neurons are given in advance to the layers.

[0033] In FIG. 3, association coefficients between the hidden layer 2 and the output layer 3 are determined such that the original input/output pattern is realized by a combination of linearly separable patterns realized by the neurons of the hidden layer 2. Specifically, the association coefficients between the hidden layer 2 and the output layer 3 are determined such that, using the positive number ε, the sum (θ + e) of association coefficients between output neurons and first through odd-numbered hidden neurons is greater than the threshold θ, and the sum (θ - e) of association coefficients between output neurons and first through even-numbered hidden neurons is smaller than the threshold θ.

[0034] The threshold function $L(\mathbf{w}^T\mathbf{x} - \theta)$ as a basis for the above mathematical model is known as the McCulloch-Pitts neuron model. Since the threshold function $L(\mathbf{w}^T\mathbf{x} - \theta)$ can be implemented by hardware or software, the neural network model shown in FIG. 3 can also be realized.

(2) Formation of linearly separable patterns:

[0035] In the neural network wherein the association coefficients (weights) between the higher neurons, i.e., between the hidden and output layers, are determined as described above in (1), association coefficients indicated by w_{ij}, i = 1, 2, ···, n; j = 1, 2, ···, k in FIG. 3) between lower neurons, i.e., between the input and hidden layers, are determined according to a learning process, described below, for generating linearly separable patterns from a linearly inseparable pattern. The linearly inseparable pattern is then transformed into linearly separable patterns by the neural network model with the association coefficients thus determined.

[Learning process]

[0036] If target and actual outputs are the same as each other, the weights are not altered.

[0037] If target and actual outputs are different from each other, then using α indicated by:

$$\alpha = (\theta - \mathbf{w}^{dT}\mathbf{x}) / \sum_i x_i$$

the weights $\mathbf{w}^d$ are altered according to the following equation:

$$w^{d+1} = w^d + \alpha^* x$$

[0038] Having effected the above learning process, it is checked whether the alteration of the association coefficients $\mathbf{w}^d$ is 0 or not with respect to all input patterns. If the alteration of the association coefficients $\mathbf{w}^d$ is 0 and there is a pattern with respect to which target and actual outputs are different from each other, then it is determined that a linearly inseparable pattern has been learned. If this is the case, then any subsequent patterns with respect to which target and actual outputs are different from each other are stored, obtaining a set of linearly inseparable elements called a "linearly inseparable core pattern set" that serves as an origin of the linearly inseparable pattern.

[0039] If a pattern to be learned is linearly separable, then it can be learned according to the above learning process. If a pattern to be learned is linearly inseparable, then a linearly inseparable core pattern set is obtained according to the above learning process. Thereafter, one of the two patterns, e.g., an OFF pattern of the two ON/OFF patterns, is selected from the obtained linearly inseparable core pattern set such that, for example, an input pattern with a maximum norm is selected. The transformation is carried out by changing the selected OFF pattern to an ON pattern.

[0040] Based on the newly obtained ON/OFF pattern, the learning process is effected again, and the allocation is finished when a certain finishing condition (described later on) is satisfied. If no finishing condition is satisfied, then a linearly inseparable core pattern set is determined, and one of the OFF patterns thereof is changed to an ON pattern. The above process is repeated until all patterns are made linearly separable.

[0041] A linearly separable pattern can be obtained according to the above learning process. The result of the process is shown in FIG. 2(c). To obtain a pattern as shown in FIG. 2(d), the difference between the pattern shown in FIG. 2 (c) and the original pattern, i.e., the pattern shown in FIG. 2 (a) is used as a new original pattern, and the above learning process is effected to check if the new original pattern is linearly separable or not.

[0042] The aforesaid process is executed until the difference between two patterns becomes a linearly separable pattern. Thereafter, the input/output patterns that have been given at first, including a linearly separable pattern, are learned with the neural network, as shown in FIG. 3, including hidden layer neurons having weights corresponding to the patterns. FIG. 3 will be described in detail later on.

[0043] The principle of the present invention is applicable not only to the three-layer neural network as shown in FIG. 3, but also to a multilayer neural network having an input layer 1, a plurality of hidden layers 2, and an output layer 3, as shown in FIG. 21. In FIG. 21, the neural network has three hidden layers 2, and neurons of the first and third hidden layers 2 as counted from the input layer 1 are generated as required as the learning process proceeds. Neurons of the second hidden layer 2 are provided in advance as output neurons (corresponding to the output neuron y in FIG. 3) with respect to the neurons of the first hidden layer 2 adjacent to the input layer 1. FIG. 21 will also be described in detail later on.

[0044] An embodiment of the present invention will now be described in greater detail by way of mathematical models, successively in topics as follows:

(1) Linear separability;
(2) Linearly separable allocation algorithm;
(3) Determination of separating hyperplanes; and
(4) Overall algorithm.

(1) Linear separability:

[0045] The input/output relationship between neurons is expressed by:

$$x_{out} = h(x^T w - \theta) \quad (1)$$

where the transformation function h is a continuous threshold function represented by:

$h(z) = 1$ when $z > 0$, and

$$h(z) = 0 \text{ when } z < 0 \quad (2)$$

where **x** is a n-dimensional binary vector (Boolean vector), indicating the state value of input neurons, x_{out} is a binary scalar, indicating the state value of output neurons, $\mathbf{w} \in R^n$ is an association weight vector of a synapse corresponding to the input vector **x**, and $\theta \in R^1$ is a neuron threshold in the form of a positive constant.

[0046] The term "learning" denotes a process in which the association weights **w** are altered until the output x_{out} of a network composed of a neuron model according to the equation (1) above has a desired value with respect to an input of m n-dimensional binary vectors **x**.

[0047] It is assumed that the set of m n-dimensional input vectors $\mathbf{x} = (x_i)$ is represented by X, a set of input vectors whose target output is 1 is referred to as an ON pattern set and indicated by X_{ON}, and a set of input vectors whose target output is 0 is referred to as an OFF pattern set and indicated by X_{OFF}. The elements of these latter two sets are called respectively as "ON pattern" and "OFF pattern". The following assumption is provided for these ON and OFF patterns:

Assumption a: $X_{ON} \cap X_{OFF} = \phi$ (empty set).

[0048] A process for determining **w** which satisfies the equations:

$$h(x^T W - \theta) = 1 \ (x \in X_{ON})$$

$$h(x^T W - \theta) = 0 \ (x \in X_{OFF}) \quad (3),$$

that is, the inequalities:

$$x^T W > \theta \ (x \in X_{ON})$$

$$x^T W < \theta \ (x \in X_{OFF}) \quad (4)$$

is the learning process.

[0049] If there is a solution **w** according to the above formulas, then there is a hyperplane:

$$x^T W = \theta \quad (5)$$

which separates the pattern sets X_{ON}, X_{OFF} in a strong sense. At this time, the sets X_{ON}, X_{OFF} are said to be linearly separable. If not, the sets X_{ON}, X_{OFF} are said to be linearly unseparable.

[0050] In the learning of a binary neural network, only linearly separable patterns can be handled between two layers insofar as the the McCulloch-Pitts neuron model is employed. To learn linearly inseparable patterns, therefore, it is necessary to use a neuron network of three or more layers, and to handle a linearly separable input/output relationship between the layers, so that a linearly inseparable input/output relationship can be achieved as a whole. It has already been proved that any optional input/output relationship can achieved by a neuron network of three or more layers.

[0051] In considering a learning algorithm for a neuron network of three or more layers, it is important how to determine a teaching signal.

[0052] According to the present embodiment, a pattern allocation process is employed to divide a linearly inseparable pattern into a plurality of linearly separable patterns and allocate and learn the linearly separable patterns between the input and hidden layers.

(2) Linearly separable allocation algorithm:

[0053] A suitable target output is given to the hidden layer for a linearly inseparable pattern to be learned. To explain such a process, a learning pattern as shown in FIG. 4 is provided.

[0054] FIG. 4 shows a linearly separable allocation by way of example. Specifically, in a general **x** (vector) space, half spaces S^k_{ON} (k = 1, ···, 5) are determined with respect to a given pattern to be learned. Spaces indicated by the arrows in FIG. 4 are representative of the regions of S^k_{ON}.

[0055] If it is assumed that black dots of those denoted by a through i are ON patterns and white dots OFF patterns in FIG. 4, then the pattern sets X_{ON}, X_{OFF} are given as follows:

$$X_{ON} = \{b, c, f, g, i\},$$

$$X_{OFF} = \{a, d, e, h\}.$$

[0056] Obviously, these pattern sets are linearly inseparable. If, however, hyperplanes (two-dimensionally, straight lines) are introduced as shown in FIG. 4 and half spaces (indicated by the arrows) represented by these hyperplanes are expressed by:

$$S^1_{ON} = \{x | w^{1T}x > 0\},$$
$$S^5_{ON} = \{x | w^{5T}x > 0\} \quad (6),$$

then whether patterns **x** are contained in these half spaces can be determined by checking if the state values of neurons that are expressed by:

$$x^1 = h(w^{1T}x - 0),$$

$$x^5 = h(w^{5T}x - 0) \quad (7)$$

can be 1 or not. The half space S^1_{ON} is realized by the first neuron x^1. While the target outputs (the separation of the OFF and ON patterns) of the patterns a, b, c, f, g, i are realized by x^1, the separation of the OFF patterns d, e, h is not realized. To realize the separation of the OFF patterns d, e, h, a difference set $S^1_{ON} - S^2_{ON}$ which is produced by subtracting S^2_{ON} from S^1_{ON} is considered.

[0057] The difference set $S^1_{ON} - S^2_{ON}$ can be determined by checking if the state value of a neuron expressed by:

$$x_{1,2} = h((\theta + \varepsilon)x^1 - 2\varepsilon x^2 - \theta)$$

according to the neuron model (1) is 1 or not. Although desired output values for the patterns a, b, c, d, e, h are obtained by $x_{1,2}$, no satisfied outputs are obtained for the patterns f, g, i. To realize the patterns f, g, i, S^3_{ON} is added, providing a set:

$$(S^1_{ON} - S^2_{ON}) \cup S^3_{ON}.$$

The state value of a neuron expressed by:

$$x_{1,3} = h((\theta + \varepsilon)x^1 - 2\varepsilon x^2 + 2\varepsilon x^3 - \theta)$$

obtained with respect to the above set realizes all the patterns except h.

[0058] If the above argument is applied to obtain desired outputs for all the patterns, then a neuron for giving a desired output y is expressed by:

$$y = h((\theta + \varepsilon)x^1 - 2\varepsilon x^2 + 2\varepsilon x^3 - 2\varepsilon x^4 + 2\varepsilon x^5 - \theta) \quad (8)$$

where $\varepsilon > 0$. The equations (7), (8) can be generalized respectively into:

$$x^k = h(w^{kT}x - 0) \tag{9}$$

and

$$y = h((\theta + \varepsilon)x^1 - 2\varepsilon x^2 + 2\varepsilon x^3 \cdots - \theta) \tag{10}$$

and can be achieved by the network as shown in FIG. 3.

[0059] FIG. 3 shows the arrangement of the neural network according to the present embodiment. The hidden layer 2 of the neural network shown in FIG. 3 has as many neurons (k neurons) as required by the algorithm.

[0060] The weight coefficients between the output layer 3 and the hidden layer 2 have a value of $\theta + \varepsilon$ only between $x^1 \sim x_{out}$, and have alternate values of -2ε and 2ε for and subsequent to x^2.

[0061] In order to realize a linearly inseparable learning pattern, there may be determined separating hyperplanes which alternately divide only ON patterns and only OFF patterns, as described above, with respect to the given learning pattern.

[0062] Once the separating hyperplanes have been determined, the association weight between the output neuron y and the first neuron x^1 of the hidden layer is made larger than the threshold θ by ε as indicated by the equation (8), and the association weights between the output neuron y and the second and following neurons x^2, x^3, ···, x^k are equalized to those having an absolute value of 2ε with alternate signs, in order to express the state in which the ON and OFF patterns are alternately divided by the hyperplanes.

[0063] If the half plane S^2_{ON} shown in FIG. 4 were directed in the opposite direction, then only x^1 is cancelled out by x^2 in its ON region, and $x_{1,2}$ has the same value as x^2 at all times.

[0064] To prevent S^2_{ON} from being determined in this way, the following relationship must be satisfied:

$$(S^{k+1}{}_{ON} \cap X) \subset (S^k{}_{ON} \cap X) \tag{11}$$

[0065] In view of the above argument and the fact that the input pattern set X is a Boolean set, the algorithm for the pattern allocation method can generally be described as follows:

Algorithm α (Linearly separable allocation algorithm):

[Step 1]

[0066] With the iteration (repetition) number being represented by k and an input vector set which will be described later on being represented by x^k, it is first assumed that:

$X^0 = X$ (input pattern set), $X^1{}_{ON} = X_{ON}$ (ON pattern set), $X^1{}_{OFF} = X_{OFF}$ (OFF pattern set), and $k = 1$.

[Step 2]

[0067] X^{k-1} is separated into a half space containing all elements of $X^k{}_{ON}$ and a half space containing at least one of only elements of $X^k{}_{OFF}$. Hyperplanes which satisfy the relationship (11), i.e., hyperplanes:

$$w^{kT}x = \theta \tag{12}$$

which satisfy the following relationship:

$$w^{kT}x > \theta, \ \forall x \in X^k{}_{ON},$$

$$w^{kT}x < \theta, \ \exists x \in X^k{}_{OFF}, \text{ and}$$

$$\{x \in X | w^{kT}x > \theta\} \subset X^{k-1} \tag{13}$$

are determined.

[0068] The set of input vectors contained in half spaces on the $X^k{}_{ON}$ sides of these hyperplanes is represented by

X^k which is given by:

$$X^k = \{x \in X | w^{kT}x > \theta\} \quad (14).$$

[Step 3]

[0069] If $X^k = X^k_{ON}$, then the algorithm is finished. If not, then

$$X^{k+1}_{ON} = X^k - X^k_{ON}, \text{ and}$$
$$X^{k+1}_{OFF} = X^k_{ON} \quad (15).$$

The iteration number k is set to k + 1, and control goes to the step 2.

[0070] The above algorithm attempts to separate the learning pattern X^{k-1} into the sets X^k_{ON}, X^k_{OFF} with the hyperplanes $\mathbf{w}^{kT}\mathbf{x} = \theta$ in each iteration. If the separation is successful, then the algorithm is finished. If not, the learning pattern X^{k-1} is separated into a half space containing all elements of X^k_{ON} and a half space containing at least one of the elements of X^k_{OFF} with the hyperplanes $\mathbf{w}^{kT}\mathbf{x} = \theta$.

[0071] For such separation, the relationship (11) must be satisfied as described above. To meet this requirement, the conditional formula (13) which is equivalent to the relationship (11) is added to the algorithm. From this condition and the equation (14) results:

$$X^{k+1} \subset X^k,$$

and the number of elements of X^k is reduced by at least one for each iteration.

[0072] The half space separated from the set X^k_{ON} by the hyperplanes $\mathbf{w}^{kT}\mathbf{x} = \theta$ contains only the elements of X^k_{OFF}, and these elements are separated from X^k by the hyperplanes. On the X^k_{ON} sides of the hyperplanes, there exist all the elements of X^k_{ON} and the remaining elements of X^k_{OFF} which are mixed together without being separated. The next iteration attempts to separate portions of X^k_{ON}, X^k_{OFF} which have not been separated in the present iteration.

[0073] At this time, not only the ON and OFF patterns are simply separated from each other, but, as described with reference to FIG. 4, hyperplanes for alternately separating the ON and OFF patterns must be determined, and X^k_{ON} must be expressed using sum and difference sets alternately as indicated by the equation (10) above. To this end, target outputs for all the elements of the set X^k_{ON} and some elements of tile set X^k_{OFF}, which have not been fully separated, are reversed, providing a learning pattern in the next iteration.

[0074] The set X^k_{ON} may be reversed by replacing X^k_{ON} with X^{k+1}_{OFF} as indicated by the second one of the equations (15). However, the reversal of the set X^k_{OFF} cannot simply be effected.

[0075] Since the elements of X^k_{OFF} in the half space that does not contain X^k_{ON} which is separated by the hyperplanes $w^{kT}x = \theta$ have already been only an OFF or ON pattern depending on the iteration, these elements are separated from the other elements of X^k on the opposite sides of the hyperplanes. These elements are considered already divided patterns in FIG. 4. Under the condition (12) above, these patterns are not contained in the ON sides of the hyperplanes that are determined in a subsequent iteration, and their outputs remain unchanged. Therefore, the elements of X^k_{OFF} within the set X^k where the above patterns have been excluded from X are reversed. The result produced by subtracting X^k_{ON} from X^k is updated as X^{k+1}_{OFF}, as indicated by the first one of the equations (15).

[0076] Through the above updating process, X^k_{ON} and X^k_{OFF} are equal to those which are produced by reversing subsets of the original ON and OFF pattern sets in each iteration, as indicated by:

$$X^k_{ON} \subset X_{ON} \text{ (k is an odd number)},$$
$$X^k_{ON} \subset X_{OFF} \text{ (k is an even number)} \quad (16)$$

where X^k corresponds to S^k_{ON} in FIG. 4, and X_{ON} is expressed by:

$$X_{ON} = (X^1 - X^2) \cup .... \cup (X^{k-1} - X^k)$$

(k is an even number).

$$X_{ON} = (X^1 - X^2) \cup .... \cup (X^k)$$

(k is an odd number),

[0077] The existence of the separating hyperplanes which satisfy the relationship (13) in the step 2 of the algorithm α is proved by the following [Theorem 1]:

[Theorem 1]:

[0078] Any single point in an n-dimensional Boolean vector set B^n is separable from the remaining set in a strong sense.

(Proof)

[0079] An optional element $\mathbf{x}^a$ is taken from the set B^n. The remaining set which is left by taking $\mathbf{x}^a$ from B^n is denoted by U. It is checked whether $\mathbf{x}^a$ can be expressed by the convex association of the elements of the set U.

[0080] It is assumed that

$$U = \{x_1, x_2, \cdots, x_q \in B^n\}.$$

[0081] If $\mathbf{x}^a$ can be expressed by the convex association of the elements of the set U, then $\mathbf{x}^a$ is represented by the following equations:

$$\mathbf{x}^a = \sum_{i=1}^{q} \mu_i \mathbf{x}_i$$

$$\sum_{i=1}^{q} \mu_i = 1 \qquad \cdots(17)$$

$$(\mu_i \geq 0,\ i = 1, \cdots, q)$$

[0082] The vectors $\mathbf{x}^a, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_q$ have elements of 0 or 1, and are all different from each other. Therefore, there exists j such that at least one of the elements $x_{1j}, x_{2j}, \cdots, x_{qj}$ of each vector has a value different from the other elements. The element of the righthand side of the first one of the equation (17):

$$\sum_{i=1}^{q} \mu_i \mathbf{x}_i$$

has a value of x_{ij}, i.e., 0 or 1, if only one μ_i is 1. However, if only one μ_i is 1, then since the vectors $\mathbf{x}^a, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_q$ are all different from each other, the first one of the equation (17) is not satisfied. If two or more μ_i are nonzero, then in order that the inequality:

$$0 < \sum_{i=1}^{q} \mu_i x_{ij} < 1 \qquad \cdots(18)$$

is not satisfied with respect to all j, but

$$\sum_{i=1}^{q} \mu_i x_i$$

has a value of 0 or 1, the vectors $\mathbf{x}^i$ with respect to μ_i which is not 0 must all be the same. This is contrary to the fact that the vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_q$ are all different from each other. Therefore, there is no μ_i, i = 1, ···, q which would satisfy the equations (17). Consequently, $\mathbf{x}^a$ is not expressed by the convex association of the elements of the set U, and is not contained in the convex hull of the set U.

[0083] It can be said from the above discussion that the convex hull of the set U does not contain $\mathbf{x}^a$. This fact, together with the following separation theorem, indicates that the convex hull of the set U and $\mathbf{x}^a$ can be separated by hyperplanes.

[Separation theorem]

[0084] It is assumed that Γ and Λ are two convex sets which are not empty, with Γ being compact and Λ being closed. If the convex sets Γ and Λ do not intersect with each other, then there exists a plane $\{x | x \in R^n, \mathbf{c}\cdot\mathbf{x} = \alpha\}$ $(\mathbf{c} \neq \mathbf{0})$ which separates the convex sets Γ and Λ from each other in a strong sense, and vice versa. Stated otherwise, the relationship:

$$\Gamma \cap \Lambda = 0 \leftrightarrow \left[\begin{array}{l} \exists \mathbf{c} \neq \mathbf{0} \text{ and} \\ \quad \alpha: \ \mathbf{x} \in \Gamma \rightarrow \mathbf{cx} < \alpha \\ \quad\quad \mathbf{x} \in \Lambda \rightarrow \mathbf{cx} > \alpha \end{array}\right]$$

is satisfied.

[0085] The fact that the convex hull of the set U and $\mathbf{x}^a$ can be separated by hyperplanes is equivalent to the fact that the set U and $\mathbf{x}^a$ can be separated by hyperplanes. However, it is generally not easy to determine the hyperplanes $\mathbf{w}^{kT}\mathbf{x} = \theta$. Instead of directly determining the hyperplanes $\mathbf{w}^{kT}\mathbf{x} = \theta$, some of the OFF patterns of the original patterns are replaced with ON patterns to define half spaces for determining these hyperplanes. These patterns must of necessity be linearly separable. According to the present invention, the above separating hyperplanes are obtained by giving these patterns to the hidden layer neurons for learning.

[0086] Inasmuch as a plurality of linearly separable patterns thus generated with respect to the original pattern are allocated to the hidden layer neurons for learning, this process is called "Linearly Separable Allocation Method", which will be referred to as "LISA" in short.

(3) Determination of separating hyperplanes:

[0087] According to Theorem 1, how separating hyperplanes which satisfy the conditions of the step 2 of the algorithm α are determined becomes the next problem (original problem), which is equivalent to the determination of $\mathbf{w}$ expressed by the relationship (4) above.

[0088] It is assumed, for example, that in the event a given pattern is not linearly separable, a pattern set which makes the given pattern linearly separable is obtained. By replacing ON and OFF patterns of the pattern set thus obtained, the pattern as a whole can be made linearly separable. In the present invention, the pattern set which makes the given pattern linearly separable is called "Linearly Unseparable Core Pattern Set", which will be referred to as "LUCPS" in short.

[0089] A "dual problem" which serves as a basis for defining LUCPS will hereinafter be described below, and then an "optimization" problem with respect to an original problem" will be derived for actually determining LUCPS. Furthermore, the relationship between optimization conditions for the optimization problem and linearly unseparable core patterns, and the process of extracting LUCPS will also be described below.

(3)-1 Dual problem:

[0090] In a preparatory step, suffix sets I_{ON}, I_{OFF} are produced from the ON pattern set X_{ON} and the OFF pattern

set X_{OFF} as follows:

$$I_{ON} = \{i | x_i \in X_{ON}\},$$

$$I_{OFF} = \{i | x_i \in X_{OFF}\}.$$

[0091] Auxiliary Theorem 1, given below, is applied to the original problem (4), deriving Theorem 2.

[Auxiliary Theorem 1]

[0092] Given a matrix $A \in R^{m*n}$ and a vector $\mathbf{b} \in R^m$,

I. there is a solution $x \in R^n$ for $Ax > b$, or
II. there is a solution $\mathbf{y} \in R^m$ for $A^T\mathbf{y} = \mathbf{0}$, $\mathbf{b}^T\mathbf{y} \geq 0$, $\mathbf{y} > \mathbf{0}$. However, the statements I, II do not hold true at the same time. $\mathbf{y} > \mathbf{0}$ indicates that all elements of $\mathbf{y}$ are 0 or more and at least one element is not 0.

(Proof)

[0093] The statement I is equivalent to

I'. there is a solution a solution $\mathbf{x} \in R^n$ ($\xi \in R$) for $\xi > 0$, $A\mathbf{x} > \mathbf{b}\xi$. If Gordon's theorem is applied to the statement I', then it is true or
II'. there is a solution $\mathbf{y}$ for $A^T\mathbf{y} = \mathbf{0}$, $(\mathbf{y}^T, \mathbf{b}^T\mathbf{y}) > \mathbf{0}$. However, the statements I', II' do not hold true at the same time.

If $\mathbf{y} = \mathbf{0}$, since $\mathbf{b}^T\mathbf{y} = 0$,
$(\mathbf{y}^T, \mathbf{b}^T\mathbf{y}) > \mathbf{0}$ has the same value as
$\mathbf{y}^T > \mathbf{0}, \mathbf{b}^T\mathbf{y}) > 0$.
Therefore, either the statement I is true or the statement II is true.

[Theorem II]

[0094] With respect to θ having a certain fixed positive value,

I. there is a solution $\mathbf{w}$ for
$\mathbf{x}_i^T\mathbf{w} > \theta$, $i \in I_{ON}$
$\mathbf{x}_i^T\mathbf{w} < \theta$, $i \in I_{OFF}$, or
II. there is a solution λ for

$$\sum_{i \in I_{ON}} \lambda_i \mathbf{x}_i = \sum_{i \in I_{OFF}} \lambda_i \mathbf{x}_i$$

$$\sum_{i \in I_{ON}} \lambda_i \geq \sum_{i \in I_{OFF}} \lambda_i, \quad \lambda \geq 0 \qquad \cdots(19)$$

However, the statements I, II do not hold true at the same time.

(Proof)

[0095] A learning pattern matrix $V \in R^{M*N}$ is generated as follows:

$$v_i = \begin{cases} x_i & (i \in I_{ON}) \\ -x_i & (i \in I_{OFF}) \end{cases} \qquad \cdots(20)$$

From vi, (i = 1, 2, ···, m), a matrix V is produced as follows:

$$V = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_m^T \end{bmatrix} \qquad \cdots(21)$$

Likewise,

$$\gamma_i = \begin{cases} \theta (i \in I_{ON}) \\ -\theta (i \in I_{OFF}) \end{cases} \qquad \cdots(22)$$

Then, the statement I is equivalent to

I'. there is a solution **w** for $V\mathbf{w} > \gamma$. From Auxiliary Theorem 1, the statement I' holds true, or
II'. there is a solution λ, for $V^T\lambda = 0$, $\gamma^T\lambda \geq 0$, $\lambda > 0$. However, the statements I', II' do not hold true at the same time.

[0096] Using the equation (21), the statement II' is rewritten as follows:

$$\sum_i \lambda_i \mathbf{v}_i = 0$$

$$\sum_i^i \lambda_i \gamma_i \geq 0$$

$$\lambda > 0$$

[0097] Furthermore, from the equations (20), (22),

$$\sum_{i \in I_{ON}} \lambda_i \mathbf{x}_i = \sum_{i \in I_{OFF}} \lambda_i \mathbf{x}_i$$

$$\sum_{i \in I_{ON}} \lambda_i \theta \geq \sum_{i \in I_{OFF}} \lambda_i \theta \qquad \cdots(23)$$

$$\lambda > 0$$

Since θ is a fixed positive number, the second one of the equations (23) becomes:

$$\sum_{i \in I_{ON}} \lambda_i \geq \sum_{i \in I_{OFF}} \lambda_i$$

Therefore, the statement I is true or the statement II is true.

[0098] Since the solutions do not exist at the same time, the problem expressed by the equations (19) is called a dual problem for the original problem (4). The solution λ to the dual problem (17) can be regarded as a positive association coefficient determined such that linear combinations of ON, OFF pattern vectors are equal to each other, and indicates that the sum of association coefficients with respect to ON patterns is equal to or greater than the sum of association coefficients with respect to OFF patterns.

[0099] The dual problem can be rewritten as follows:

$$\left.\begin{aligned} c\sum_{i\in I_{ON}}\mu_i x_i &= \sum_{i\in I_{OFF}}\mu_i x_i \\ \sum_{i\in I_{ON}}\mu_i &= \sum_{i\in I_{OFF}}\mu_i = 1 \\ \mu &\geq 0,\quad c \geq 1 \end{aligned}\right\} \qquad \cdots(24)$$

[0100] The equations (24) show that they have a solution μ when some of the convex associations of the ON pattern set X_{ON} are contained in a cone which is made up of the convex associations of the OFF pattern set X_{OFF} and the origin, as schematically shown in FIG. 5.

[0101] FIGS. 5(a) and 5(b) show the meaning of a dual problem in an x space. If convex associations of an ON pattern set are contained in a cone made up of the origin and the convex associations of an OFF pattern set, then the dual problem has a solution and the original problem has no solution. In FIG. 5(a), the dual problem has no solution, and the original problem has a solution. In FIG. 5(b), the dual problem has a solution, and the original problem has no solution.

[0102] It can be seen from FIGS. 5(a) and 5(b) that linear separability is determined by the relative positional relationship between all patterns. However, as shown in FIGS. 6(a) and 6(b), with respect to a linearly inseparable pattern set, an original pattern can be transformed into a linearly separable pattern by replacing some ON and OFF patterns with each other in the linearly inseparable pattern set.

[0103] FIGS. 6(a) and 6(b) illustrate transformation of a linearly inseparable pattern. In FIG. 6(a), a cone made up of the origin and the convex associations of an OFF pattern set contain convex associations of an ON pattern set. When some (two upper left points) of the patterns of the OFF pattern set of the cone are changed to ON patterns as shown in FIG. 6(b), the cone no longer contains any ON patterns.

[0104] The fact that some convex associations of the ON pattern set X_{ON} are contained in the cone made up of the origin and the convex associations of the OFF pattern set X_{OFF} in FIGS. 5(a), 5(b), 6(a), and 6(b) indicates that there is a solution μ to the problem (18), and the solution μ corresponds to the solution λ to the problem (17). The solution λ is an association coefficient in the case where the linear combinations of the elements of X_{ON}, X_{OFF} are the same as each other. If the components of the association coefficient λ can be divided into 0 components and positive components, then the positive components are considered to determine the solution to the dual problem. Thus, the positive components of λ are involved in linear separability of the original problem.

[0105] Generally, if the number of patterns is larger than the number of dimensions, then all the patterns are linearly dependent, and the solution λ to the dual problem for most of the patterns can be positive. However, since the component value of a Boolean vector is 0 or 1, the dimensions of a space defined by the patterns is smaller than patterns of continuous values. This does not apply if the number of learning patterns is close to the number of all combinations that can be possible with original input/output patterns, though such a case does not generally occur.

[0106] If the positive component of a solution to a dual problem is determined, therefore, a linearly separable pattern set X^{k-1} determined in the step 2 of the algorithm α should be obtained by changing OFF patterns to ON patterns. Hidden layers for achieving, through a learning process, the linearly separable pattern set X^{k-1} thus obtained, and output neurons for combining those hidden layers are determined to reconstruct the original patterns according to the algorithm α, for thereby achieving the original linearly separable patterns.

[0107] Based on the foregoing, a "linearly inseparable core pattern set" which is a pattern set as a basis for linearly inseparability is defined as follows:

[Definition 1]

[0108] A set I_{LU} of patterns with respect to nonzero elements of the solution $\lambda = [\lambda_i]$ to the dual problem (17) is defined as a linearly inseparable core pattern set (LUCPS) as follows:

$$I_{LU} = \{i | \lambda_i > 0\}$$

[0109] It is possible to determine the solution to a dual problem by applying a condition $\lambda > \mathbf{0}$ to a sweeping-out method which checks if simultaneous equations are solvable or not. As the number of problem dimensions and the number of patterns increase, however, a combination explosion takes place.

[0110] Consequently, a more practical method will hereinafter be described.

(3) - 2 Optimization problem with respect to the original problem:

[0111]

[0112] An optimization problem to determine a linearly inseparable core pattern set I_{LU} is formulated below.

[0113] For speeding up convergence, the following transformation functions are introduced into ON and OFF patterns:

$$\left.\begin{aligned} h_{ON}(z) &= \begin{cases} 1, & z \geq 0 \\ 1 + z, & z < 0 \end{cases} \\ h_{OFF}(z) &= \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases} \end{aligned}\right\} \quad \cdots (25)$$

[0114] These functions are represented as shown in FIGS. 7 (a) and 7 (b) , and cannot be differentiated when z = 0.

[0115] h_{ON} is provided by modifying the transformation function according to the equations (2), and a portion thereof where the argument is positive, i.e., the original problem is admissible is equal to h. A portion where the argument is negative, i.e., the original problem is inadmissible is linear.

[0116] In sigmoid functions, a gradient exists only in the vicinity of a region where the argument is 0. In the above transformation functions, a constant gradient is present at all times in a portion where the original problem is inadmissible. Therefore, irrespective of the value of the association weight $\mathbf{w}^k$, stable and quick convergence can be achieved in the learning process. For the same reasons, h_{OFF} is expressed according to the second one of the equations (25).

[0117] The transformation functions h_{ON}, h_{OFF} are allocated to the ON and OFF patterns, as described above, and the patterns are individually considered. By maximizing h_{ON}, there are determined weights that realize the ON patterns, and by maximizing h_{OFF}, there are determined weights that realize the OFF patterns. The following optimization problem will be considered in order to determine the weights for realizing the ON and OFF patterns:

$$\max_{\mathbf{w}} \left(\sum_{x \in X_{ON}} h_{ON}(\mathbf{x}^T\mathbf{w} - \theta) - \sum_{x \in X_{OFF}} h_{OFF}(\mathbf{x}^T\mathbf{w} - \theta) \right) \quad \cdots (26)$$

[0118] With the sums of h_{ON}, - h_{OFF} being maximized, the solution search is conducted toward regions where the output is 1 with respect to the ON patterns, and also toward regions where the output is 0 with respect to the OFF patterns. If the input pattern is linearly separable, then the solution of the next problem gives an admissible solution to the original problem (4). The formula (26) is a maximization problem for the function:

$$\phi(\mathbf{w}) = \sum_{x \in X_{ON}} h_{ON}(\mathbf{x}^T\mathbf{w} - \theta) - \sum_{x \in X_{OFF}} h_{OFF}(\mathbf{x}^T\mathbf{w} - \theta) \quad \cdots (27)$$

Since h_{ON} is a concave function and h_{OFF} is a convex function, ϕ is a concave function with respect to **w**, and the local maximum of ϕ is equal to a global maximum. The problem (20) can be solved by a gradient method for an undifferentiable

optimization problem.

(3) - 3 Relationship between an optimization condition and a dual problem:

[0119]

[0120] An optimization condition is determined from an optimum solution to the optimization problem (20), and it will be indicated that the coefficient of the optimization condition is equal to the solution to a dual problem. it can be seen that if the original problem (4) is linearly inseparable, a pattern of nonzero (positive) coefficients of an optimization condition equation with respect to the optimum solution to the problem (20), i.e., LUCPS, can be determined.

[0121] The optimization condition for the problem (20) will first be described below.

[0122] The function ϕ is a partially undifferentiable function, and has no gradient in that it is undifferentiable. However, it has a hypogradient or a general gradient under certain conditions. The definition of a hypogradient and an optimization condition for an optimization problem will be described below.

[0123] In the optimization problem

$$\max_{\mathbf{x}} \; f(\mathbf{x}) \qquad \cdots (28)$$

regarding $x \in R^n$ of a concave function f, if the concave function f is undifferentiable with respect to $\mathbf{x}_0$, the hypogradient is a set of **z** which satisfies:

$$f(x) \leq f(x_0) + z^T(x - x_0), \; \forall x \in X$$

Using that set $\partial f(\mathbf{x}_0)$, the optimization condition for the optimization problem is expressed by:

$$0 \in \partial f(x_0)$$

[0124] As the functions h_{ON}, - h_{OFF} are concave in a one-dimensional space R^1, there is obviously a hypogradient of ϕ in $\mathbf{w} \in \text{int}X$. Using a hypogradient set $\partial\phi$ of ϕ, the optimization condition for optimizing an undifferentiable concave function can be expressed as follows:

$$0 \in \partial\phi(w^0) \qquad (29)$$

where $\mathbf{w}^0$ represents an optimum solution.

If

$$\phi_{ON}(w;x)=h_{ON}(xTw-\theta),$$

and

$$\phi_{OFF}(w;x)=h_{OFF}(xTw-\theta),$$

then

$$\partial\phi(w) =$$

$$\{ \sum_{x \in X_{ON}} \nabla\phi_{ON}(w;x) - \sum_{x \in X_{OFF}} \nabla\phi_{OFF}(w;x) \mid$$

$$\nabla\phi_{ON}(w;x) \in \partial\phi_{ON}(w;x),$$

$$\nabla\phi_{OFF}(w;x) \in \partial\phi_{OFF}(w;x) \} \quad \cdots (30)$$

where

$$\nabla\phi_{ON}(w;x) = \{0\} \quad (x^T w > \theta)$$

$$\nabla\phi_{ON}(w;x) = co\{0,x\} \quad (x^T w = \theta)$$

$$\nabla\phi_{ON}(w;x) = \{x\} \quad (x^T w < \theta)$$

$$\nabla\phi_{OFF}(w;x) = \{x\} \quad (x^T w > \theta)$$

$$\nabla\phi_{OFF}(w;x) = co\{0,x\} \quad (x^T w = \theta)$$

$$\nabla\phi_{OFF}(w;x) = \{0\} \quad (x^T w < \theta) \quad (31)$$

co indicates a convex hull defined as follows:

$$co\{0,x\} = \{\lambda x | 0 \le \lambda \le 1\} \quad (32)$$

From the equations (31), (32),

$$\partial\phi_{ON}(w;x_i) = \{\lambda_i x_i\} \quad (i \in I_{ON})$$

$$\partial\phi_{OFF}(w;x_i) = \{\lambda_i x_i\} \quad (i \in I_{OFF}) \quad (33)$$

where
with respect to $i \in I_{ON}$,

$$\lambda_i = 0 \quad (x^T w > \theta)$$

$$\lambda_i = [0,1] \quad (x^T w = \theta)$$

$$\lambda_i = 1 \quad (x^T w < \theta)$$

with respect to $i \in I_{OFF}$,

$$\lambda_i = 1 \quad (x^T w > \theta)$$

$$\lambda_i = [0,1] \qquad (x^T w = \theta)$$

$$\lambda_i = 0 \qquad (x^T w < \theta) \tag{34}$$

From the equations (30), (33), (34),

$$\sum_{i \in I_{ON}} \lambda_i x_i - \sum_{i \in X_{OFF}} \lambda_i x_i \in \partial\phi(w) \qquad \cdots(35)$$

From the equations (29) and (35), there is λ_i which satisfies the equations (34) and the following equation:

$$\sum_{i \in I_{ON}} \lambda_i x_i - \sum_{i \in I_{OFF}} \lambda_i x_i = 0 \qquad \cdots(36)$$

Furthermore, $0 \le \lambda_i \le 1$ is satisfied with respect to all i's.

[0125] If the original problem (4) is linearly separable, then

$$\partial\phi_{ON}(w;x_i) = \{0\},\ i \in I_{ON}$$

$$\partial\phi_{OFF}(w;x_i) = \{0\},\ i \in I_{OFF}$$

in the optimum solution $\mathbf{w}^0$ to the problem (20).

[0126] If the original problem (4) is linearly inseparable, then since an inadmissible pattern exists, the hypogradient set $\partial\phi_{ON}$ or $-\partial\phi_{OFF}$ has nonzero elements as can be understood from the equations (34). Therefore, there exists nonzero λ_i which satisfies the equation (36). That is,

$$\lambda > 0 \tag{37}$$

[0127] In order to introduce the relationship between λ which satisfies the optimization condition formula and the dual problem (17), the following theorem is obtained:

[Theorem 3]:

[0128] If there is no admissible solution **w** to the original problem (4):

$$x_i^T w - \theta > 0,\ i \in I_{ON}$$

$$x_i^T w - \theta < 0,\ i \in I_{OFF}$$

and if the following equation is satisfied:

$$\sum_{i \in I_{OFF}} \lambda_i x_i - \sum_{i \in I_{ON}} \lambda_i x_i = 0,\ \lambda > 0$$

then the following relationship is satisfied:

$$\sum_{i \in I_{ON}} \lambda_i \geq \sum_{i \in I_{OFF}} \lambda_i$$

(Proof)

[0129] The foregoing is apparent from the following Auxiliary Theorem 2:

If there is no solution **x** for $A\mathbf{x} > \mathbf{b}$ with respect to a given matrix $A \in R^{m*n}$, then the solution $\mathbf{y} \in R^p$ for

$$A^T y = 0, \; y > 0$$

satisfies

$$b^T y \geq 0$$

(Proof)

[0130] When there is a solution $\mathbf{y}^0$ to

$$A^T y = 0, \; y > 0$$

$\mathbf{b}^T\mathbf{y}^0 < 0$ which is a negation of $\mathbf{b}^T\mathbf{y}^0 \geq 0$ is assumed, introducing a contradiction.

[0131] Since the fact that there is no solution **x** for $A\mathbf{x} > \mathbf{b}$ is equivalent to the fact that there is a solution **y** for

$$A^T y = 0, \; y > 0, \; b^T y \geq 0,$$

there is no $\mathbf{y}^0$ for

$$A^T y^0 = 0, \; y^0 > 0, \text{ and } b^T y^0 < 0$$

[0132] If there is no admissible solution to the original problem (4), i.e., if the learning patterns X_{ON}, X_{OFF} are linearly unseparable, then, from the equations (19), (34), (36), (37) and Theorem 3,

$$\Gamma \subset \Lambda$$

where Γ is a set of λ which satisfies the optimization condition equation (36) for the optimization problem (20) and the expression (37) and Λ is a solution set for the dual problem (17). Since the solution to the dual problem (17) satisfies the optimization condition equation (36) and the expression (37), the following relationship is satisfied:

$$\Gamma \supset \Lambda$$

Therefore,

$$\Gamma = \Lambda$$

Consequently, in the case where the original problem is linearly unseparable, then λ that is determined from the optimum solution to and the optimum condition for the optimization problem (20) is equal to the solution to the dual problem.

(3)-4 Extraction of a linearly unseparable core pattern:

[0133] It can be seen from the above theory that if the original problem is linearly inseparable, then λ that satisfies the equations (31) with respect to the solution $\mathbf{w}^0$ to the optimization problem (20) is determined, and the positive component of λ corresponds to a linearly inseparable core pattern that corresponds to $\mathbf{w}^0$. However, the problem (20) may not necessarily have an undifferentiable point, and an optimum solution may not always be determined for the problem (20). It will be shown below that LUCPS (linearly inseparable core pattern set) can be extracted even when no optimum solution is determined.

[0134] An algorithm for solving the optimization problem (20) will be described below.

Algorithm β (algorithm for solving the problem (20)):

[Step 1]

[0135] A certain initial point $\mathbf{w}^1$ is appropriately selected, and an iteration number d is set to d = 1.

[Step 2]

[0136] $h_{ON}(\mathbf{x}^T\mathbf{w}^d-\theta)$; $\mathbf{x} \in X_{ON}$ and $h_{OFF}(\mathbf{x}^T\mathbf{w}^d-\theta)$; $\mathbf{x} \in X_{OFF}$ are calculated. If there exists even one pattern $\mathbf{x}_p$ which satisfies:

$$h_{ON}(x_p{}^T w^d-\theta) \neq 1;\ x_p \in X_{ON},$$

or

$$h_{OFF}(x_p{}^T w^d-\theta) \neq 1,\ x_p \in X_{OFF}$$

then control proceeds to a next step. If not, $\mathbf{w}^d$ is regarded as the solution $\mathbf{w}^0$ to the problem (20), and the algorithm is finished.

[Step 3]

[0137] With respect to p in the step 2, the following correction is made:

$$w^{d+1} = w^d + \alpha^p \Delta w^d(x_p) \tag{38}$$

where

$$\alpha^p = \frac{\theta - x_p \bar{w}^d}{\sum_{i=1}^{n} x_{pi}} \qquad \cdots(39)$$

$$\Delta w^d(x_p) = \nabla_w \phi_{ON}(w;x_p),\ x_p \in X_{ON}$$

$$\Delta w^d(x_p) = \nabla_w \phi_{OFF}(w;x_p),\ x_p \in X_{OFF}$$

where $\nabla_w \phi_{ON}(\mathbf{w};\mathbf{x}_p)$ and $\nabla_w \phi_{OFF}(\mathbf{w};\mathbf{x}_p)$ are expressed as follows:

$$\nabla_w \phi_{ON}(w;x_p) = x_p,$$

$$\nabla_w \phi_{OFF}(w;x_p) = x_p \quad (40)$$

[Step 4]

[0138] The algorithm is finished if either one of the following finishing conditions a, d, e is satisfied. If not, then $\mathbf{w}^d \leftarrow \mathbf{w}_o$ when the iteration number $d = d_0$ where d_0 is a certain number. When $d > d_0$, the algorithm is finished if the finishing condition b is satisfied.
Otherwise, the algorithm is finished if the finishing condition c is satisfied. When the iteration number is not in any of these cases, the iteration number d is updated by $d \leftarrow d + 1$, and control goes back to the step 2. The finishing conditions a through e are defined as follows:
(Finishing condition a) A sufficiently large positive integer j_0 is determined, and the algorithm is finished if there is j which satisfies:

$$|w^d - w^{d-j}| < \zeta$$

with respect to $j = 1, \cdots, j_0$ where ζ is a sufficiently small positive number.
(Finishing condition b) The algorithm is finished if $\mathbf{w}^d = \mathbf{w}_0$ when $d > d_0$.
(Finishing condition c) The algorithm is finished when $d > d_0$. d_0 is a positive integer which is somewhat larger than the maximum iteration number at the time an n-dimensional linearly separable problem is solved.
(Finishing condition d) The algorithm is finished if a hypogradient set $\partial\phi(\mathbf{w}^d)$ of a function ϕ with respect to $\mathbf{w}^d$ contains a zero vector.
(Finishing condition e) The algorithm is finished if $\phi(\mathbf{w}^d) < \phi(\mathbf{w}^{d-1})$.

[0139] The finishing conditions are most difficult to determine in the algorithms used in the present invention. Whether an actual pattern is linearly separable or not is determined based on the finishing conditions. The finishing conditions a, b serve to check if a change $\Delta\mathbf{w}^d$ in $\mathbf{w}$ in one cycle or more becomes $\mathbf{0}$ or not when the value d becomes larger than a certain value d_0, i.e., when the convergence has sufficiently been in process. The finishing condition d is used to halt the algorithm when an actual pattern cannot be determined as to whether it is linearly inseparable or not, based on the finishing conditions a, b. The learning time in the present invention is governed by whether the finishing conditions are good or bad. In a numerical example described later on, $d_0 = 3$.

[0140] The method of determining the step width according to the equation (39) is based on the idea that a minimum width required is given so as to make an inadmissible pattern x_p at a present point $\mathbf{w}^d$ admissible with respect to the original problem.

[0141] The transformation functions h_{ON}, h_{OFF} are essentially undifferentiable at 0, and a hypogradient must be given thereto as indicated by the equations (31). However, it is generally difficult to positively determine a hypogradient set. To avoid the difficulty, a hypogradient is given according to the equations (40) in view of the nature of the neuron transformation function.

[0142] If original input/output patterns are linearly separable, i.e., if the original problem (4) has an admissible solution, there is determined a solution $\mathbf{w}^0$ for achieving all input/output patterns in the same manner as the learning process for perceptrons, according to the algorithm β. if original input/output patterns are linearly inseparable, a λ set obtained from the optimization condition for the problem (20) is equal to a solution set for a dual problem. The path of $\mathbf{w}^d$ converges up to near an optimum solution that satisfies the optimization condition equation (36) for the problem (2) and the expression (37), but does not stop at the optimum solution, and vibrates in or around the optimum solution. Even if an optimum solution is obtained, it is difficult to positively determine a hypogradient set of an object function of the problem (20). Therefore, it is also difficult to determine whether $\mathbf{w}^d$ satisfies the optimization condition.

[0143] The algorithms according to the present invention does not need an optimum solution, but may determine LUCPS (linearly inseparable core pattern set). In most cases, even if all LUCPSs are not obtained, a subset thereof may be obtained and an OFF pattern to be transformed may be selected from the subset.

[0144] According to the algorithm β, $\mathbf{w}^d$ vibrates in the vicinity of an optimum solution with respect to a sufficiently large d ($d > d_0$). In the vicinity of an optimum solution, an inadmissible component of the original problem (4) is considered to correspond to a linearly inseparable core pattern.

[0145] According to the method (32) of determining a corrected step width with respect to an inadmissible pattern, the step width with respect to the present inadmissible pattern is determined such that an output for at least the present pattern is equalized to a target output.

[0146] Then, interference occurs between linearly inseparable core patterns, and $\mathbf{w}^d$ vibrates between regions of $\mathbf{w}$ which satisfy a target output between these patterns. Thus, if $\mathbf{w}^d$ starts to vibrate in the vicinity of an optimum solution

after it has converged to a certain extent, then components or patterns in the equation (4) which are made inadmissible subsequently are considered to be an element of LUCPS. The LUCPS should be obtained by registering these patterns.

[0147] A suitable OFF pattern is selected from the obtained pattern set, and changed to an ON pattern, thus reducing the number of elements of the LUCPS.

[0148] If the conditional expression (13) of the algorithm for the linearly separable allocation method described above under (2) is not satisfied, then any reduction in the number of elements of X^k is not assured. Selection of an OFF pattern to be transformed in order to satisfy the above condition is made according to the following Rule A or Rule B:

[Rule A]

[0149] An OFF pattern which is most distant from the origin shall be selected.

[Rule B]

[0150] An OFF pattern whose inner product with $\mathbf{w}^d$ is largest shall be selected.

[0151] Linearly separable patterns are obtained by transforming patterns which cause linear inseparability, one by one. Separating hyperplanes for the linearly separable allocation algorithm are determined from the obtained linearly separable patterns.

[0152] The above process may be incorporated in the linearly separable allocation algorithm α.

[0153] FIGS. 8 and 9 are flowcharts of a learning method according to the present invention.

[0154] As shown in FIG. 8, first, input patterns (vectors) $\mathbf{x}_p$ and output patterns (scalars) t_p (p is the number of m patterns) are given in a step 1. Then, a target output pattern $\mathbf{t}$ is set to an initial value $\mathbf{t}^1$ in a step 2, and an iteration number k is set to 1 in a step 3. A target output pattern $\mathbf{t}^k$ with respect to a kth hidden neuron is substituted in a linearly separable output pattern $\mathbf{t}^k_M$ in a step 4, and a transformation routine is carried out in a step 5. The transformation routine will be described later with reference to FIG. 9.

[0155] Thereafter, a differential pattern which is produced by subtracting $\mathbf{t}^k$ from $\mathbf{t}^k_M$ is given as a (k+1)th output pattern $\mathbf{t}^{k+1}$ in a step 6, and the steps 4 through 6 are repeated until $\mathbf{t}^{k+1} = \mathbf{0}$. Specifically, after the step 6, a step 7 determines whether $\mathbf{t}^{k+1} = \mathbf{0}$. If $\mathbf{t}^{k+1} \neq \mathbf{0}$, then the iteration number k is incremented by 1 in a step 8, and control returns to the step 4.

[0156] When $\mathbf{t}^{k+1}=\mathbf{0}$, the neutral network shown in FIG. 2 is constructed with the weights (association coefficients) determined as described above between output and hidden layers, in a step 9.

[0157] The transformation routine will be described below.

[0158] As shown in FIG. 9, first a weight $\mathbf{w}^1$ is set to $\mathbf{0}$ in a step 11. Then, a pattern number p and the iteration number d of the algorithm β are set to 1, and a variable sw indicative of a wrong answer (described later) is set to 0 in a a step 12. A step 13 determines whether the linearly separable target output $\mathbf{t}^k_M$ agrees with an actual output h $(\mathbf{x}_p{}^T\mathbf{w}^d-\theta)$ with respect to a pattern p. If not, then the weight $\mathbf{w}^d$ is altered or corrected according to the equations (38) and (39), and the variable sw is set to 1 in a step 14. Thereafter, a step 15 determines whether $\mathbf{w}^d$ is vibrating ($d > d^0$) or not. If vibrating, the pattern p is registered in a step 16.

[0159] The steps 13 through 16 are executed with respect to m patterns. Specifically, after the pattern p is registered, or if the linearly separable target output $\mathbf{t}^k_M$ agrees with an actual output h $(\mathbf{x}_p{}^T\mathbf{w}^d-\theta)$, a step 17 determines whether $p \leq m$ or not. If $p \leq m$, then p and d are incremented by 1 in a step 18, and control goes back to the step 13.

[0160] When p exceeds m, a step 19 determines whether either one of the finishing conditions a ~ e is satisfied or not. If satisfied, then the pattern p is registered as a linearly unseparable pattern in a step 20. One OFF pattern is selected from the registered patterns as a pattern p' in a step 21. A target output $\mathbf{t}^k_{Mp'}$ for the selected pattern p' is set to 1 in a step 22, from which the processing goes back to the step 12.

[0161] If none of the finishing conditions are satisfied in the step 19, then a step 23 determines whether the variable sw = 0 or not. If not, control returns to the step 12, and if yes, the pattern is registered as linearly separable in a step 24.

(4) Algorithm for the linearly separable allocation method (LISA):

(4)-1 Algorithm:

[0162] The linearly separable allocation algorithm α described above under (2) and the algorithm β for solving the optimization problem described under (3)-4 are joined to each other, making algorithms γ, δ^k as described below.

Algorithm γ:

[Step 1]

[0163] $X^1_{ON} = X_{ON}$, $X^1_{OFF} = X_{OFF}$. The iteration number k is set to k = 1.

[Step 2]

[0164]

$$X^k = X^k_{ON}.$$

[Step 3]

[0165] Using the algorithm δ^k (described later on), it is checked whether X^k_{ON}, $X^k - X^k_{ON}$ are linearly separable or not.

[Step 4]

[0166] If linearly separable, then $\mathbf{w}^o$ obtained by the algorithm δ^k is set to $\mathbf{w}^k$, and the processing goes to a step 5. If linearly inseparable, then an element $\mathbf{x}_m$ whose norm is maximum is selected from an intersection of sets $\{\mathbf{x}_i | \mathbf{x}_i \in I_{LU}\}$, $\{X^k - X^k_{ON}\}$ obtained according to the algorithm δ^k. After $X^k \leftarrow X^k \cup \{\mathbf{x}_m\}$, control goes back to the step 4.

[Step 5]

[0167] If $X^k = X^k_{ON}$, then control goes to a step 6. If not, then

$$X^{k+1}_{ON} \leftarrow X^k + X^k_{ON},$$

$$X^{k+1}_{OFF} \leftarrow X^k_{ON}$$

and the iteration number k is updated into k + 1. The processing then goes to the step 2.

[Step 6]

[0168] The three-layer neural network model expressed by the equations (9), (10) is constructed, and the algorithm γ is ended.

Algorithm δ^k:

[Step 1]

[0169] A certain initial point $\mathbf{w}^1$ is selected. $Y^1_{ON} = X^k_{ON}$, $Y^1_{OFF} = X^k - X^k_{ON}$, and the linearly unseparable core pattern set $I_{LU} = \Phi$ (empty set) . The iteration number d is set to d = 1.

[Step 2]

[0170] $h_{ON}(\mathbf{x}^T\mathbf{w}^d - \theta)$ $[\mathbf{x} \in Y^k_{ON}]$ and $h_{OFF}(\mathbf{x}^T\mathbf{w}^d - \theta)[\mathbf{x} \in Y^k_{OFF}]$ are calculated. If even one pattern p exists which satisfies:

$$h_{ON}(x_p{}^T w^d - \theta) \neq 1 \ (x_p \in Y^k_{ON}),$$

or

$$h_{OFF}(x_p{}^T w^d - \theta) \neq 0 \ (x_p \in Y^k_{OFF}),$$

then control goes to a next step. If not, $\mathbf{w}^d$ is set to $\mathbf{w}^0$, and the algorithm is ended.

[Step 3]

[0171] All patterns p in the step 2 are corrected according to the equations (38), (39). If $d > d_0$, then the patterns p are added to the linearly inseparable core pattern set I_{LU}. That is,

$$I_{LU} \leftarrow I_{LU} \cup \{p\}$$

d_0 is a maximum value for the convergence iteration number at the time there is a solution to a predetermined original problem.

[Step 4]

[0172] If any of the finishing conditions a, d, e is satisfied, then the algorithm is finished. If not satisfied, when $d = d_0$, $\mathbf{w}^d \leftarrow \mathbf{w}_o$. If the finishing condition b is satisfied, then the algorithm is finished. If not, then the algorithm is finished when the finishing condition c is satisfied. If the finishing condition c is not satisfied, then $d \leftarrow d + 1$, and the processing goes back to the step 2.

(4)-2 Speeding-up of algorithm:

[0173] Addition of improvements, described below, to the algorithms makes it possible to speed up the learning process.

1. The weights are calculated as integer-type weights. It is important that the threshold be not too small compared to the number of input neurons. If the threshold were too small, the iteration number would be unnecessarily be increased due to a quantization error, possibly causing infinite recurrence.
2. In the step 3 of the algorithm γ, before executing the algorithm δ^k, j which satisfies

$$x_j \geq x_i$$

with respect to $i \in I_{ON}$, $j \in I_{OFF}$ changes from OFF patterns to ON patterns.

[0174] The significance of the speeding-up process under 2 above is as follows:
This speeding-up process is effective to reduce the number of times that the algorithm δ^k is carried out to discover a linearly unseparable core pattern set I_{LU}. However, since other patterns than I_{LU} may be rendered ON, the algorithm δ^k may possibly be subject to recurrence. No recurrence took place with a four-input pattern in an experiment described later on.

[0175] FIGS. 10(a) through 10(c) show a transformation for the speeding-up process. White dots indicate OFF patterns, and black dots ON patterns. As shown in FIG. 10(a), those OFF patterns (on the righthand or upper side of dotted lines) which are equal to or larger than the ON patterns are transformed into ON patterns. Thereafter, as shown in FIG. 10(b), the ON and OFF patterns are clearly separated from each other. However, as shown in FIG. 10(c), there is a transformation in which the number of changed patterns is smaller.

[0176] When there are an OFF pattern set and a ON pattern as shown in FIG. 10(a), since a line interconnecting the origin and the ON pattern passes through the convex hull of the OFF pattern set, these patterns are linearly unseparable. According to the above speeding-up process, all OFF patterns on the lefthand or upper side of the dotted lines are transformed into ON patterns, as shown in FIG. 10(b), and become linearly separable. This transformation is not a minimum transformation for making linearly separable patterns. The transformation may be less if effected as shown in FIG. 10(c).

[0177] FIGS. 11 and 12 show the processes according to the algorithms γ, δ^k in greater detail. FIG. 13 illustrates the definition of letters and symbols in FIGS. 11 and 12. FIGS. 14 through 16 are a detailed flowchart of the algorithm γ including the speeding-up process shown in FIGS. 10(a) through 10(c). FIG. 17 is a detailed flowchart of the algorithm δ^k. FIGS. 14 through 17 are a specific representation of the processes shown in FIGS. 8 and 9. The steps shown in FIGS. 14 through 17 are basically the same as those shown in FIGS. 8 and 9, and will not be described in detail.

(4)-3 Expansion of the algorithms:

[0178] According to the above algorithms, the neuron threshold θ is of a fixed positive value. Therefore, $h(\mathbf{w}^T\mathbf{x}-\theta)$

necessarily becomes 0 at the origin **0**. when all inputs are 0, the output has a value of 0. To achieve all patterns including those in which all inputs are 0, a bias neutron 4 may be added as shown in FIG. 18. The bias neuron 4 is always ON so that the output is ON even when all the neurons of the input layer 1 are OFF. The neural network arrangement shown in FIG. 18 makes a learning process possible even if all inputs have a value of 0.

[0179] In actual biological environments, such a neuron is of a circuit arrangement as shown in FIG. 19 which has a synaptic coupling for self feedback. If such a neuron is used, all input/output patterns including those in which all inputs are 0 can be learned according to the above algorithms.

[0180] FIG. 19 shows a bias neuron unit with a self feedback loop. Once an ON input signal is applied to the bias neuron unit, the bias neuron unit keeps an ON value at all times. The value of the bias neuron unit may be made OFF if an inhibitory neuron is connected to the input thereof.

[0181] The neural network for executing the above algorithms basically has multiple inputs and a single output. The neural network may be expanded into a multiple-input multiple-output neural network as shown in FIG. 20. Since synaptic couplings between output and hidden layers 3, 2 do not interconnect all the neurons in these layers, the learning process can be carried out at high speed.

[0182] The present invention is not limited to the above three-layer network structures, but may be applied to a multilayer neural network structure having a plurality of hidden layers 2 as shown in FIG. 21.

[0183] The neural network shown in FIG. 21 comprises an input layer 1, an output layer 3, and three hidden layers 2 . The hidden neurons having state values $x^1{}_1, \cdots, x^{k1}{}_1; x^1{}_2, \cdots, x^{k2}{}_2; \cdots; x^1{}_p, \cdots, x^{kp}{}_p$ in the first hidden layer from the input layer 1, and hidden neurons having state values $2_{x_1}, \cdots, 2_{x_r}$ in the third hidden layer adjacent to the output layer 3 are generated as required as the learning process progresses. The hidden neurons having state values $x_1', \cdots, x_p'$ in the second hidden layer are provided in advance as output neurons (corresponding to the output neuron having a state value y in FIG. 3) with respect to the hidden neurons in the first hidden layer.

<Experiments>

[0184] To ascertain the effectiveness of LISA, it was checked as to whether it can learn all patterns including linearly inseparable patterns. A numerical experiment was also been conducted to see if LISA causes an NP (nondeterministic polynomial) problem or not. The results of these experiments were compared with those of learning processes according to back propagation. The experiments employed "EWS,SUN4/260" manufactured by Sun Micro Systems.

1. Inspection of the learning capability of a neural network:

[0185] Generally, it is impossible to determine whether a neural network can learn all patterns actually. A four-input, one-output neural network was experimented to see if it can learn all input/output patterns. Since the LISA has only a positive threshold, the output would be 0 when all inputs are 0 unless the neural network is expanded into a multiple-input multiple-output neural network as shown in FIG. 20. The learning pattern included two outputs (0, 1) with respect to 15 input patterns except for an input pattern in which all four inputs are 0. The number of all input/output patterns was $2^{15} = 32768$.

[0186] A back-propagation neural network used for comparison had two layers except for an input layer, and included 4 neurons in the input layer, 8 neurons in the hidden layer, and one neuron in the output layer. Learning parameters were a learning step width $\eta = 0.9$ and a bias term coefficient $\alpha = 0.9$. The learning process according to back propagation compared a target output and an output pattern of the network in each iteration, and ended when all of the compared outputs were the same. The output of the network was converted into a binary signal using a threshold of 0.5 for comparison with the target output. Therefore, if the output was 0.6, a binary value of 1 was compared with the target output, and if the output was 0.3, a binary value of 0 was compared with the target output.

[0187] When the iteration number reached 3,000, the learning of the patterns was interrupted as no solution that would satisfy all the patterns was considered to exist. The results of the experiments are given in Table 1 below.

Table 1

Learning of 32,768 patterns		
Algorithm	Back propagation	LISA
Calculation time	162,530.4 sec.	81.9 sec.
Wrong answers	4085	0
Percentage of correct answers	88 %	100 %

[0188] As can be seen from Table 1, the percentage of correct answers according to the back propagation process was 88 %, whereas the percentage of correct answers according to LISA was 100 %. The time required for learning the patterns according to the back propagation process was about 1,984 times the time required for learning the patterns according to LISA. When the iteration number reached 3,000 according to the back propagation process, the learning process was interrupted as producing wrong answers. The calculation times according to LISA and the back propagation process while no wrong answers were produced are given in Table 2 below. The results given in Table 2 were obtained when 100 patterns were checked as they were solved by the back propagation process, i.e., learned in learning cycles less than 3,000 learning cycles.

Table 2

Learning of 100 patterns		
Algorithm	Back propagation	LISA
Calculation time	102.8 sec.	0.2 sec.
Wrong answers	0	0
Percentage of correct answers	100 %	100 %

[0189] Table 2 indicates that subject to no wrong answers, the LISA is 514 times faster than the back propagation process. Therefore, even if the back propagation process can learn patterns with the iteration number being less than 3000, the LISA is about 500 times faster than the back propagation process. However, as described above with respect to the speeding-up of the algorithms, the learning speed is lower for integer calculations if the threshold is too small. The threshold should preferably be of a value which is as large as possible depending on the memory capacity of the computer used and the dimensions of problems. For four-dimensional problems, the maximum number of neurons generated in the hidden layers of the LISA was 5 during a trial performance on 32, 768 patterns.

2. Experiment to see if the LISA causes an NP problem:

[0190] Even though the LISA is faster than the back propagation process, if it causes an NP problem when the number of input patterns increases, then the calculation time exponentially increases, and will not be practical. An experiment was conducted to see how the calculation time of the LISA varies with respect to the number of patterns to be learned.

[0191] The patterns to be learned were 7-input, 1-output patterns, and N patterns determined according to uniform random numbers were learned 100 patterns each time. The results are shown in the logarithmic graph of FIG. 22. The graph was approximated with a polynomial ($aN^b + c$) with respect to each of pattern numbers 0, 25, 75, resulting in illustrated curves. The polynomial was represented by $1.5 \times 10^{-4}N^{2.99} + 0.05$. As the pattern number approaches 90, the gradient of the equation was larger than the experimental values. For 7 inputs, the calculation time of the LISA was of a polynomial order, and does not cause an NP problem. For seven-dimensional 127 patterns, the maximum number of neurons generated in the hidden layers of the LISA was 22 during a trial performance on 100 random patterns.

[0192] FIG. 22 also shows the calculation time of the back propagation process. In this experiment, the back propagation process was modified as follows:

When the iteration number exceeded 300, it was considered that the learning process would not converge any more, and the association weights were initialized again according to uniform random numbers, and the learning process was started again. This procedure was repeated up to 5 times. It can be seen from this graph that when the pattern number is 40 or higher, the LISA is about 100 times faster than the back propagation process.

[0193] In FIG. 22, the LISA (represented by white dots) is about 100 times faster than the back propagation process (black dots). Since the graph has a vertical axis representing the logarithm of the time, the amount of calculation of either algorithm did not exponentially increase.

[0194] If the calculation time increased exponentially, then any plotted calculation time would be indicated by a straight line having a certain gradient. The results of the LISA and the back propagation process show that the gradients thereof were reduced as the pattern number increased. The percentage of correct answers of the back propagation process suddenly started to decrease when the pattern number exceeded about 40 as shown in FIG. 23. Since the curve was similar to that of a sigmoid function, it was approximated by a sigmoid function as shown. The percentage of the LISA was kept at 100 % irrespective of the number of patterns. It indicates that the LISA could learn all input/output patterns generated by uniform random numbers.

[0195] As shown in FIG. 23, the percentage of correct answers of the LISA was kept at 100 % regardless of the pattern number increasing. The percentage of correct answers of the back propagation decreased at the rate of a

sigmoid function as the pattern number increased.

[0196] The results of the experiments indicate the following:

1. The LISA does not cause an NP problem with respect to the pattern number.
2. For seven inputs, the LISA is about 100 times faster than the back propagation process.
3. The number of patterns that can be learned (the percentage of correct answers) by the back propagation process decreases at the rate of a sigmoid function as the pattern number increases.

[0197] When the number of patterns is 50 or more, the percentage of correct answers of the back propagation greatly decreases, and the actual learning speed of the LISA appears to be much more faster than the back propagation process.

[0198] As described above, the learning algorithm for a binary neural network which can learn even linearly unseparable patterns according to the present invention makes it possible for the neural network to learn all four-input one-output patterns except for a pattern with all inputs being zero. The learning algorithm of the invention is capable of learning patterns much faster than the conventional algorithm, and requires a calculation time on a polynomial order.

[0199] With the present invention, as described above, a linearly unseparable input/output pattern is transformed into several linearly separable patterns, which are then combined by an output layer so that they produce the same output as the original input/output pattern. Therefore, a neural network can learn all input/output patterns irrespective of whether they are linearly separable or unseparable.

[0200] Since hidden layer neurons are generated as required, if a pattern is linearly separable, one neuron is sufficient, and if a pattern is linearly unseparable, a minimum number required of neurons or a close number of neurons are sufficient. Thus, a memory required to achieve desired input/output patterns may be of a minimum capacity required. Since the number of association weights between neurons to be modified is also minimum, the learning speed is very high.

[0201] Since the conventional back propagation process uses a sigmoid function as a neuron transformation function, the weights vary to a large degree as a local optimum point is approached, but to a very small degree at a point far from the local optimum point. The transformation function used in the present invention has a constant gradient anywhere at points that are inadmissible for the learning process. Because the step width is of a minimum making the present patterns admissible, the association coefficients converge at high speed irrespective of the distance from an optimum point.

[0202] According to the present invention, since hidden layer neurons are automatically generated as required, there is no concern required over how many hidden layer neurons are to be employed.

[0203] The analysis of a neural network that is produced according to a learning process is highly complex and difficult to achieve. According to the present invention, the network as shown in FIG. 3 is constructed according to the linearly separable allocation method. Since the network shown in FIG. 3 achieves calculations between sets as indicated by the equations described with reference to FIG. 2(b), it can easily be analyzed.

[0204] Although certain preferred embodiments of the present invention have been shown and described in detail, it should be understood that various changes and modifications may be made therein without departing from the scope of the appended claims.

Claims

1. A neural network comprising:

an input layer having a plurality of input neurons for receiving an input signal said input signal containing an input/output pattern set (P), said plurality of input neurons comprising a predetermined fixed number of input neurons;

one or more hidden layers having one or more hidden neurons for processing a signal received from, said plurality of input neurons, and

an output layer having one or more output neurons for processing a signal received from said hidden neurons and for producing an output signal,

said input layer and said hidden layers being coupled to each other by association coefficients (wji) determined by a learning process,

said hidden layers and said output layer being coupled to each other by association coefficients (wji) determined such that a sum of association coefficients between said output layer neurons and first through odd-numbered sequential hidden neurons in a single hidden layer is larger than a predetermined threshold and a sum of association coefficients between said output neurons and first through even-numbered sequential hid-

den neurons in the single hidden layer is smaller than said predetermined threshold,
the neural network automatically generating the hidden neurons of said single hidden layer as required according to a predetermined process as said learning process progresses,
wherein said given input/output pattern set is implemented by a combination of linearly separable pattern sets realized by said hidden neurons, each hidden neuron corresponding to a respective one of said linearly separable patterns said neural network being characterized in that said learning process is effected by modifying association coefficients w^d, between said hidden layers and said input layer according to the formula

$$w^{d+1} = w^d + \alpha^* x$$

where

d is an index indicative of a number of repeated calculations, and
α, a bias coefficient, is indicated by

$$\alpha = (\theta - w^{dT}x) / \sum_i x_i$$

w^d, w^{d+1} being weights, $w \in R^n x$ being the input pattern or set of input vectors
T indicating transposition, and
θ being a neuron threshold value, and $\theta \in R^1$

if target and actual outputs are different from each other,
extracting a set of patterns from all given patterns according to a predetermined pattern set extracting process, and
transforming one by one patterns causing inseparability selected from said extracted set of patterns according to a predetermined rule into patterns of different type, the learning process being effectuated again until a certain finishing condition is satisfied, thereby finally obtaining a linearly separable pattern set;
determining said association coefficients between said output layer and said generated hidden neurons so as to realize said given input/output pattern set by means of a combination of linearly separable pattern sets realized by said hidden neurons.

2. The neural network according to claim 1, wherein said one or more hidden layers comprise hidden neurons, wherein the hidden neurons in an odd-numbered hidden layer from said input layer are generated as required as said learning process progresses, and the hidden neurons in an even-numbered hidden layer from said input layer are provided in advance as output neurons with respect to the hidden neurons in said odd-numbered hidden layer.

3. The neural network according to claim 1, further comprising a bias neuron connected to each of said hidden neurons, said bias neuron continuously being in an ON state when an input signal for making said bias neuron ON is applied.

4. The neural network according to claim 1, wherein said hidden neurons are divided into a plurality of groups, respective ones of said output neurons being provided as coupled to respective ones of the groups of the hidden neurons resulting in a multiple-input, multiple-output network.

5. A learning method for a neural network comprising an input layer having a plurality of input neurons for receiving an input signal said input signal containing an input/output pattern set (P), one or more hidden layers having one or more hidden neurons for processing a signal received from said plurality of input neurons, and an output layer having one or more output neurons for processing a signal received from said hidden neurons and for producing an output signal, the learning method comprising the steps of:

determining whether said given input/output pattern is linearly separable or not;
applying input values from said input/output pattern to said input layer and a corresponding output values from said input/output pattern to said hidden neurons to effect a predetermined learning process on said hidden neurons, if said given input/output pattern set is linearly separable;
determining association coefficients between said output layer and said hidden layers such that a signal from

said hidden neurons and an output signal from said output neurons which receive said signal from said hidden neurons are equal to each other;
allocating a learning pattern set determined by a predetermined learning pattern set determining process, between said hidden layers and said input layer to effect said learning process on said hidden neurons, if said given input/output pattern set is linearly inseparable;
generating the hidden neurons of at least one of said hidden layers depending on said given input/output pattern set according to a predetermined process as said learning process progresses, each hidden neuron corresponding to a respective one of said linearly separable pattern sets;
said learning method being characterised in that said learning process is effected by modifying association coefficients between said hidden layers and said input layer according to the formula

$$w^{d+1} = w^{d} + \alpha^{*}x$$

where

d is an index indicative of a number of repeated calculations, and
α, a bias coefficient, is indicated by

$$\alpha = (\theta - w^{dT}x) / \sum_{i} x_i$$

w^d, w^{d+1} being weights, $w \in R^n$
x being the input pattern or set of input vectors
T indicating transposition, and
θ being a neuron threshold value, and $\theta \in R^1$,

if target and actual outputs are different from each other,
extracting a set of patterns from, all given patterns according to a predetermined pattern set extracting process, and
transforming one by one patterns causing inseparability selected from said extracted set of patterns according to a predetermined rule into patterns of different type, the learning process being effectuated again until a certain finishing condition is satisfied, thereby finally obtaining a linearly separable pattern set;
determining said association coefficients between said output layer and said generated hidden neurons so as to realize said given input/output pattern set by means of a combination of linearly separable pattern sets realized by said hidden neurons.

6. The learning method according to claim 5, wherein said predetermined rule is defined to select a pattern remotest from the origin of a coordinate space in which the pattern is presented.

7. The learning method according to claim 5, wherein said predetermined rule is defined to select a pattern whose inner product with the weight of a predetermined hidden neuron is maximum or minimum.

8. The learning method according to claim 5, further comprising the steps of:

applying an input pattern to said hidden neurons to determine whether an output of said output neurons agrees with the target output or not; and
correcting the association coefficients between said hidden layers and said input layer until a predetermined finishing condition is satisfied, if the output of said output neurons disagrees with the target output.

9. The learning method according to claim 8, wherein said finishing condition is satisfied when the number of times that said association coefficients are corrected exceeds a predetermined value.

10. The learning method according to claim 8, wherein said finishing condition is based on a comparison between a present weight and a weight which satisfies a predetermined condition at at least one time in the past, when repetitive calculations are effected in said learning process.

11. The learning method according to claim 8, wherein said association coefficients are corrected by a step width which is determined according to a predetermined correction rule each time said association coefficients are to be corrected.

12. The learning method according to claim 5, wherein said learning process is effected by employing different transformation functions h_{ON} h_{OFF} with respect to a given ON pattern whose target output is 1 and a given OFF pattern whose target output is 0, maximizing a sum of the transformation functions with respect to all patterns, and transforming said input/output pattern into a linearly separable pattern based on an optimization condition equation for maximizing the sum of the transformation functions, it said input/output pattern is linearly unseparable.

13. The learning method according to claim 5, further comprising the steps of:

determining a learning pattern according to a predetermined process from an original input/output pattern and an input/output pattern realized by hidden neurons which have been learned; and
allocating the learning pattern to hidden neurons that have not been learned.

14. The learning method according to claim 5, wherein said pattern is transformed by transforming an OFF pattern whose target output is 0 into an ON pattern whose target output is 1, and modifying the OFF pattern into an ON pattern when all elements of the OFF pattern have a value equal to or greater than the value of corresponding elements of an ON pattern.

15. The learning method according to claim 5, wherein said set of patterns is extracted by checking a change in the association coefficients between said hidden layers and said input layer, and extracting a set of patterns in which the association coefficients do not change and the target output disagrees with the actual output.

16. The learning method according to claim 8, wherein said finishing condition is based on either a value of a predetermined object function having an argument comprising said association coefficients, or a hypogradient of said object function.

17. The learning method according to claim 5, wherein said association coefficients between said output layer and said generated hidden neurons are determined so as to have alternate sign in order of the generation of said generated hidden neurons.

Patentansprüche

1. Neuronales Netzwerk, umfassend:

eine Eingabeschicht mit einer Mehrzahl von Eingabeneuronen zum Empfang eines Eingabesignals, wobei das Eingabesignal eine Eingabe/Ausgabemustermenge (P) enthält, wobei die Mehrzahl von Eingabeneuronen eine vorbestimmte feste Anzahl von Eingabeneuronen umfaßt;
eine oder mehrere versteckte Schichten mit einem oder mehreren versteckten Neuronen zum Verarbeiten eines von der Mehrzahl von Eingabeneuronen empfangenen Signals, und
eine Ausgabeschicht mit einem oder mehreren Ausgabeneuronen zum Verarbeiten eines von den versteckten Neuronen empfangenen Signals und zum Erzeugen eines Ausgabesignals,
wobei die Eingabeschicht und die versteckten Schichten miteinander durch Zuordnungskoeffizienten (w_i) gekoppelt sind, die durch einen Lernprozeß bestimmt sind,
wobei die versteckten Schichten und die Ausgabeschicht, die durch Zuordnungskoeffizienten (w_i) miteinander gekoppelt sind, derart bestimmt sind, daß eine Summe von Zuordnungskoeffizienten zwischen den Ausgabeschicht-Neuronen und ersten bis ungeradzahligen aufeinanderfolgenden versteckten Neuronen in einer einzelnen versteckten Schicht größer ist als ein vorbestimmter Schwellenwert und eine Summe von Zuordnungskoeffizienten zwischen den Ausgabeneuronen und ersten bis geradzahligen aufeinanderfolgenden versteckten Neuronen in der einzelnen versteckten Schicht kleiner ist als der vorbestimmte Schwellenwert,
wobei das neuronale Netzwerk mit fortschreitendem Lernprozeß die versteckten Neuronen der einzelnen versteckten Schicht nach Bedarf gemäß einem vorbestimmten Prozeß automatisch erzeugt,
wobei die gegebene Eingabe/Ausgabemustermenge durch eine Kombination von linear trennbaren Mustermengen implementiert wird, die durch die versteckten Neuronen realisiert sind, wobei jedes versteckte Neuron einem entsprechenden der linear trennbaren Muster entspricht,

wobei das neuronale Netzwerk dadurch gekennzeichnet ist, daß der Lernprozeß erfolgt durch Modifizieren von Zuordnungskoeffizienten (w^d) zwischen den versteckten Schichten und der Eingabeschicht gemäß der Formel

$$w^{d+1} = w^d + \alpha * x,$$

wobei

d ein Index ist, der eine Anzahl wiederholter Berechnungen bezeichnet, und
α ein Vorspann-Koeffizient ist, definiert durch

$$\alpha = (\theta - w^{dT}x) / \sum_i x_i$$

wobei w^d, w^{d+1} Gewichte sind, $w \in R^n$
x die Eingabemuster oder die Menge von Eingabevektoren ist,
T die Transposition bezeichnet und
θ ein Neuronenschwellenwert ist, und $\theta \in R^1$,

wenn sich Soll- und Istausgaben voneinander unterscheiden,
Extrahieren einer Mustermenge aus allen gegebenen Mustern gemäß einem vorbestimmten Mustermengen-Extraktionsprozeß und
einzelweises Transformieren von eine Untrennbarkeit bewirkenden Mustern, die aus der extrahierten Mustermenge gewählt sind, gemäß einer vorbestimmten Regel in Muster unterschiedlichen Typs, wobei der Lernprozeß erneut durchgeführt wird, bis eine bestimmte Beendigungsbedingung erfüllt ist, um schließlich eine linear trennbare Mustermenge zu erhalten;
Bestimmen der Zuordnungskoeffizienten zwischen der Ausgabeschicht und den erzeugten versteckten Neuronen, um mittels einer Kombination von durch die versteckten Neuronen realisierten linear trennbaren Mustermengen die gegebene Eingabe/Ausgabemustermenge zu realisieren.

2. Neuronales Netzwerk nach Anspruch 1, wobei die eine oder die mehreren versteckten Schichten versteckte Neuronen aufweisen, wobei die versteckten Neuronen mit fortschreitendem Lernprozeß in einer ungeradzahligen versteckten Schicht von der Eingabeschicht nach Bedarf erzeugt werden, und wobei die versteckten Neuronen in einer geradzahligen versteckten Schicht von der Eingabeschicht vorab als Ausgabeneuronen in bezug auf die versteckten Neuronen in der ungeradzahligen versteckten Schicht vorgesehen werden.

3. Neuronales Netzwerk nach Anspruch 1, ferner umfassend ein Vorspann-Neuron, das mit jedem der versteckten Neuronen verbunden ist, wobei das Vorspann-Neuron kontinuierlich in einem EIN-Zustand ist, wenn ein Eingabesignal zum EIN-Schalten des Vorspann-Neurons angelegt wird.

4. Neuronales Netzwerk nach Anspruch 1, wobei die versteckten Neuronen in eine Mehrzahl von Gruppen unterteilt sind, wobei jeweilige der Ausgabeneuronen mit jeweiligen der Gruppen von versteckten Neuronen gekoppelt sind, um ein Mehrfach-Eingabe-, Mehrfach-AusgabeNetzwerk zu erhalten.

5. Lernverfahren für ein neuronales Netzwerk, umfassend eine Eingabeschicht mit einer Mehrzahl von Eingabeneuronen zum Empfang eines Eingabesignals, wobei das Eingabesignal eine Eingabe/Ausgabemustermenge (P) enthält, eine oder mehrere versteckte Schichten mit einem oder mehreren versteckten Neuronen zum Verarbeiten eines von der Mehrzahl von Eingabeneuronen empfangenen Signals, sowie eine Ausgabeschicht mit einem oder mehreren Ausgabeneuronen zum Verarbeiten eines von den versteckten Neuronen empfangenen Signals und zum Erzeugen eines Ausgabesignals, wobei das Lernverfahren die Schritte umfaßt:

Bestimmen, ob das gegebene Eingabe/Ausgabemuster linear trennbar ist oder nicht;
Anlegen von Eingabewerten von dem Eingabe/Ausgabemuster an die Eingabeschicht und eines entsprechenden Ausgabewerts von dem Eingabe/Ausgabemuster an die versteckten Neuronen, um einen vorbestimmten Lernprozeß an den versteckten Neuronen zu bewirken, wenn die gegebene Eingabe/Ausgabemustermenge

linear trennbar ist;
Bestimmen von Zuordnungskoeffizienten zwischen der Ausgabeschicht und den versteckten Schichten derart, daß ein Signal von den versteckten Neuronen und ein Ausgabesignal von den Ausgabeneuronen, die das Signal von den versteckten Neuronen empfangen, zueinander gleich sind;
Zuordnen einer Lernmustermenge, die durch einen vorbestimmten Lernmustermengen-Bestimmungsprozeß bestimmt ist, zwischen den versteckten Schichten und der Eingabeschicht, um den Lernprozeß der versteckten Neuronen zu bewirken, wenn die gegebene Eingabe/Ausgabemustermenge linear untrennbar ist;
Erzeugen der versteckten Neuronen von zumindest einer der versteckten Schichten in Abhängigkeit von der gegebenen Eingabe/Ausgabemustermenge gemäß einem vorbestimmten Prozeß mit fortschreitendem Lernprozeß, wobei jedes versteckte Neuron einer entsprechenden der linear trennbaren Mustermengen entspricht, wobei das Lernverfahren dadurch gekennzeichnet ist, daß der Lernprozeß erfolgt durch Modifizieren von Zuordnungskoeffizienten zwischen den versteckten Schichten und der Eingabeschicht gemäß der Formel

$$w^{d+1} = w^d + \alpha * x,$$

wobei

d ein Index ist, der eine Anzahl wiederholter Berechnungen bezeichnet, und
α ein Vorspann-Koeffizient ist, definiert wird durch

$$\alpha = (\theta - w^{dT}x) / \sum_1 x_1$$

wobei w^d, w^{d+1} Gewichte sind, $w \in R^n$
x die Eingabemuster oder die Menge von Eingabevektoren ist,
T die Transposition bezeichnet und
θ ein Neuronenschwellenwert ist, und $\theta \in R^1$,

wenn Soll- und Istausgaben voneinander unterschiedlich sind,

Extrahieren einer Mustermenge aus allen gegebenen Mustern gemäß einem vorbestimmten Mustermengen-Extraktionsprozeß, und
einzelweises Transformieren von eine Untrennbarkeit bewirkenden Mustern, die aus der extrahierten Mustermenge gewählt sind, gemäß einer vorbestimmten Regel in Muster unterschiedlichen Typs, wobei der Lernprozeß erneut durchgeführt wird, bis eine bestimmte Beendigungsbedingung erfüllt ist, um schließlich eine linear trennbare Mustermenge zu erhalten;
Bestimmen der Zuordnungskoeffizienten zwischen der Ausgabeschicht und den erzeugten versteckten Neuronen, um mittels einer Kombination von durch die versteckten Neuronen realisierten linear trennbaren Mustermengen die gegebene Eingabe/Ausgabemustermenge zu realisieren.

6. Lernverfahren nach Anspruch 5, wobei die vorbestimmte Regel definiert ist, um ein Muster zu wählen, das am weitesten vom Ursprung des Koordinatenraums entfernt ist, in dem sich das Muster befindet.

7. Lernverfahren nach Anspruch 5, wobei die vorbestimmte Regel definiert ist, um ein Muster zu wählen, dessen inneres Produkt mit dem Gewicht eines vorbestimmten versteckten Neurons maximal oder minimal ist.

8. Lernverfahren nach Anspruch 5, das ferner die Schritte aufweist:

Anlegen eines Eingabemusters an die versteckten Neuronen zur Bestimmung, ob eine Ausgabe der Ausgabeneuronen mit der Sollausgabe übereinstimmt oder nicht; und
Korrigieren der Zuordnungskoeffizienten zwischen den versteckten Schichten und der Eingabeschicht, bis eine vorbestimmte Beendigungsbedingung erfüllt ist, wenn die Ausgabe der Ausgabeneuronen mit der Sollausgabe nicht übereinstimmt.

9. Lernverfahren nach Anspruch 8, wobei die Beendigungsbedingung erfüllt ist, wenn die Häufigkeit, mit der die

Zuordnungskoeffizienten korrigiert werden, einen vorbestimmten Wert überschreitet.

10. Lernverfahren nach Anspruch 8, wobei die Beendigungsbedingung auf einem Vergleich zwischen einem vorhandenen Gewicht und einem Gewicht beruht, das zumindest einmal in der Vergangenheit einer vorbestimmten Bedingung genügt, wenn in dem Lernprozeß repetitive Berechnungen durchgeführt werden.

11. Lernverfahren nach Anspruch 8, wobei die Zuordnungskoeffizienten um eine Schrittweite korrigiert werden, die gemäß einer vorbestimmten Korrekturregel bestimmt wird, jedesmal, wenn die Zuordnungskoeffizienten zu korrigieren sind.

12. Lernverfahren nach Anspruch 5, wobei der Lernprozeß erfolgt durch die Verwendung unterschiedlicher Transformationsfunktionen h_{EIN} h_{AUS} in bezug auf ein gegebenes EIN-Muster, dessen Sollausgabe 1 ist, sowie ein gegebenes AUS-Muster, dessen Sollausgabe 0 ist, Maximieren einer Summe der Transformationsfunktionen in bezug auf alle Muster und Transformieren der Eingabe/ Ausgabemuster in ein linear trennbares Muster auf der Basis einer Optimierungsbedingungsgleichung zum Maximieren der Summe der Transformationsfunktionen, wenn das Eingabe/Ausgabemuster linear untrennbar ist.

13. Lernverfahren nach Anspruch 5, das ferner die Schritte aufweist:

    Bestimmen eines Lernmusters gemäß einem vorbestimmten Prozeß aus einem ursprünglichen Eingabe/Ausgabemuster und einem Eingabe/Ausgabemuster, das durch die versteckten Neuronen realisiert ist, an denen erlernt wurde; und
    Zuordnen des Lernmusters zu den versteckten Neuronen, an denen nicht erlernt wurde.

14. Lernverfahren nach Anspruch 5, wobei das Muster transformiert wird durch Transformieren eines AUS-Musters, dessen Sollausgabe 0 ist, in ein EIN-Muster, dessen Sollausgabe 1 ist, und Modifizieren des AUS-Musters in ein EIN-Muster, wenn alle Elemente der AUS-Muster einen Wert haben, der gleich oder größer als der Wert der entsprechenden Elemente eines EIN-Musters ist.

15. Lernverfahren nach Anspruch 5, wobei die Mustermenge extrahiert wird durch Prüfen einer Änderung der Zuordnungskoeffizienten zwischen den versteckten Schichten und der Eingabeschicht, und Extrahieren einer Mustermnge, in der die Zuordnungskoeffizienten sich nicht ändem und die Sollausgabe mit der tatsächlichen Ausgabe nicht übereinstimmt.

16. Lernverfahren nach Anspruch 8, wobei die Beendigungsbedingung entweder auf einem Wert einer vorbestimmten Zielfunktion mit einem die Zuordnungskoeffizienten aufweisenden Argument oder auf einem Hypogradienten der Zielfunktion beruht.

17. Lernverfahren nach Anspruch 5, wobei die Zuordnungskoeffizienten zwischen der Ausgabeschicht und den erzeugten versteckten Neuronen so bestimmt werden, daß sie abwechselnde Vorzeichen haben, um die erzeugten versteckten Neuronen zu erzeugen.

Revendications

1. Réseau neuronal comportant :

    une couche d'entrée ayant une pluralité de neurones d'entrée pour recevoir un signal d'entrée, ledit signal d'entrée contenant un ensemble de modèles d'entrée/sortie (P), ladite pluralité de neurones d'entrée comportant un nombre fixe prédéterminé de neurones d'entrée,
    une ou plusieurs couches cachées ayant un ou plusieurs neurones cachés pour traiter un signal reçu depuis ladite pluralité de neurones d'entrée, et
    une couche de sortie ayant un ou plusieurs neurones de sortie pour traiter un signal reçu depuis lesdits neurones cachés et pour produire un signal de sortie,
    ladite couche d'entrée et lesdites couches cachées étant reliées les unes aux autres par des coefficients d'association (W_{ji}) déterminés par un processus d'apprentissage,
    lesdites couches cachées et ladite couche de sortie étant reliées les unes aux autres par des coefficients d'association (W_{ji}) déterminés de sorte qu'une somme de coefficients d'association entre lesdits neurones de

couche de sortie et des neurones cachés séquentiels ayant des numéros impairs à partir du numéro 1 dans une couche cachée unique est plus grande qu'un seuil prédéterminé, et de sorte qu'une somme de coefficients d'association entre lesdits neurones de sortie et des neurones cachés séquentiels ayant des numéros pairs à partir du numéro 1 dans la couche cachée unique est plus petite que ledit seuil prédéterminé,
le réseau neuronal générant automatiquement les neurones cachés de ladite couche cachée unique tel que cela est nécessaire conformément à un processus prédéterminé à mesure que ledit processus d'apprentissage progresse,
dans lequel ledit ensemble de modèles d'entrée/sortie donné est mis en oeuvre par une combinaison d'ensembles de modèles linéairement séparables réalisés par lesdits neurones cachés, chaque neurone caché correspondant à un modèle respectif desdits modèles linéairement séparables
ledit réseau neuronal étant caractérisé en ce que ledit processus d'apprentissage est effectué en modifiant des coefficients d'association (W^d) entre lesdites couches cachées et ladite couche d'entrée conformément à la formule

$$W^d{+}1 = W^d + \alpha * x$$

où
d est un indice représentatif d'un nombre de calculs répétés, et
α, un coefficient de décalage, est indiqué par

$$\alpha = (\theta - W^{dT} \cdot x) / \sum_i x_i$$

W^d, W^{d+1} étant des poids, $W \in R^n$
x étant le modèle d'entrée ou l'ensemble de vecteurs d'entrée
T indiquant une transposition, et
θ étant une valeur de seuil de neurone, et $\theta \in R^1$
si les sorties cible et réelle sont différentes l'une de l'autre,
en extrayant un ensemble de modèles à partir de tous les modèles donnés conformément à un processus d'extraction d'ensemble de modèles prédéterminés, et
en transformant un par un des modèles entraînant une nature inséparable sélectionnée parmi ledit ensemble extrait de modèles conformément à une règle prédéterminée en des modèles de type différent, le processus d'apprentissage étant effectué à nouveau jusqu'à ce qu'une certaine condition de finition soit satisfaite, de manière à obtenir en fin de compte un ensemble de modèles linéairement séparables,
en déterminant lesdits coefficients d'associations entre ladite couche de sortie et lesdits neurones cachés générés de manière à réaliser ledit ensemble de modèles d'entrée/sortie donné par l'intermédiaire d'une combinaison d'ensembles de modèles linéairement séparables réalisés par lesdits neurones cachés.

2. Réseau neuronal selon la revendication 1, dans lequel ladite couche ou lesdites plusieurs couches cachées comportent des neurones cachés, les neurones cachés dans une couche cachée de numéro impair par rapport à ladite couche d'entrée étant générés lorsque cela est nécessaire à mesure que ledit processus d'apprentissage progresse, et les neurones cachés dans une couche cachée de numéro pair par rapport à ladite couche d'entrée sont fournis à l'avance en tant que neurones de sortie par rapport aux neurones cachés de ladite couche cachée de numéro impair.

3. Réseau neuronal selon la revendication 1, comportant de plus un neurone de décalage relié à chacun desdits neurones cachés, ledit neurone de décalage étant en continu dans un état passant lorsqu'un signal d'entrée destiné à rendre passant ledit neurone de décalage est appliqué.

4. Réseau neuronal selon la revendication 1, dans lequel lesdits neurones cachés sont divisés en pluralité de groupes, des neurones respectifs parmi lesdits neurones de sortie étant fournis en étant reliés à des neurones respectifs des groupes des neurones cachés ayant pour résultat un réseau à multiples entrées et multiples sorties.

5. Procédé d'apprentissage pour un réseau neuronal comportant une couche d'entrée ayant une pluralité de neurones d'entrée pour recevoir un signal d'entrée, ledit signal d'entrée contenant un ensemble de modèles d'entrée/sortie

(P), une ou plusieurs couches cachées ayant un ou plusieurs neurones cachés pour traiter un signal reçu depuis ladite pluralité de neurones d'entrée, et une couche de sortie ayant un ou plusieurs neurones de sortie pour traiter un signal reçu depuis lesdits neurones cachés et pour produire un signal de sortie, le procédé d'apprentissage comportant les étapes consistant à :

déterminer si ledit modèle d'entrée/sortie donné est linéairement séparable ou non,
appliquer des valeurs d'entrée depuis ledit modèle d'entrée/sortie dans ladite couche d'entrée et une valeur de sortie correspondante depuis ledit modèle d'entrée/sortie dans lesdits neurones cachés pour effectuer un processus d'apprentissage prédéterminé sur lesdits neurones cachés, si ledit ensemble donné de modèles d'entrée/sortie est linéairement séparable,
déterminer des coefficients d'association entre ladite couche de sortie et lesdites couches cachées de sorte qu'un signal provenant desdits neurones cachés et un signal de sortie provenant desdits neurones de sortie qui reçoivent ledit signal provenant desdits neurones cachés sont égaux l'un à l'autre,
disposer un ensemble de modèles d'apprentissage, déterminé par un processus de détermination d'ensemble de modèles d'apprentissage prédéterminé, entre lesdites couches cachées et ladite couche d'entrée pour effectuer ledit processus d'apprentissage sur lesdits neurones cachés, si ledit ensemble donné de modèles d'entrée/sortie est linéairement inséparable,
générer les neurones cachées d'au moins une desdites couches cachées en fonction dudit ensemble donné de modèles d'entrée/sortie conformément à un processus prédéterminé à mesure que ledit processus d'apprentissage progresse, chaque neurone caché correspondant à un ensemble respectif desdits ensembles de modèles linéairement séparables,
ledit procédé d'apprentissage étant caractérisé en ce que ledit processus d'apprentissage est effectué en modifiant les positions d'association entre lesdites couches cachées et ladite couche d'entrée conformément à la formule

$$W^{d}+1 = W^{d}+\alpha^{*}x$$

où
d est un indice représentatif d'un nombre de calculs répétés, et
α un coefficient de décalage, est indiqué par

$$\alpha = (\theta - W^{dT}.x) / \sum_{i} x_i$$

w^d, w^{d+1} étant des poids, $W \in R^n$
x étant le modèle d'entrée ou l'ensemble de vecteurs d'entrée
T indiquant une transposition, et
θ étant une valeur de seuil de neurone, et $\theta \in R^1$,
si les sorties cible et réelle sont différentes l'une de l'autre,
en extrayant un ensemble de modèles à partir de tous les modèles donnés conformément à un processus d'extraction d'ensemble de modèles prédéterminés, et
en transformant un par un des modèles entraînant une nature inséparable sélectionnée parmi ledit ensemble extrait de modèles conformément à une règle prédéterminée en modèles de type différent, le processus d'apprentissage étant effectué à nouveau jusqu'à ce qu'une certaine condition de finition soit satisfaite, de manière à obtenir en fin de compte un ensemble de modèles linéairement séparables,
déterminer lesdits coefficients d'association entre ladite couche de sortie et lesdits neurones cachés générés de manière à réaliser ledit ensemble donné de modèles d'entrée/sortie par l'intermédiaire d'une combinaison d'ensembles de modèles linéairement séparables réalisés par lesdits neurones cachés.

6. Procédé d'apprentissage selon la revendication 5, dans lequel ladite règle prédéterminée est définie pour sélectionner un modèle le plus éloigné de l'origine d'un espace de coordonnées dans lequel le modèle est présenté.

7. Procédé d'apprentissage selon la revendication 5, dans lequel ladite règle prédéterminée est définie pour sélectionner un modèle dont le produit intérieur par le poids d'un neurone caché prédéterminé est maximal ou minimal.

8. Procédé d'apprentissage selon la revendication 5, comportant de plus les étapes consistant à :

appliquer un modèle d'entrée auxdits neurones cachés pour déterminer si une sortie desdits neurones de sortie correspond à la sortie cible ou non, et
corriger les coefficients d'association entre lesdits couches cachées et ladite couche d'entrée jusqu'à ce qu'une condition de finition prédéterminée soit satisfaite, si la sortie desdits neurones de sortie ne correspond pas à la sortie cible.

9. Procédé d'apprentissage selon la revendication 8, dans lequel ladite condition de finition est satisfaite lorsque lesdits coefficients d'association sont corrigés un nombre de fois qui dépasse une valeur prédéterminée.

10. Procédé d'apprentissage selon la revendication 8, dans lequel ladite condition de finition est basée sur une comparaison entre un poids présent et un poids qui satisfait à une condition prédéterminée au moins une fois dans le passé, lorsque des calculs répétitifs sont effectués dans ledit processus d'apprentissage.

11. Procédé d'apprentissage selon la revendication 8, dans lequel lesdits coefficients d'association sont corrigés par une largeur de pas qui est déterminée conformément à une règle de correction prédéterminée à chaque fois que lesdits coefficients sont à corriger.

12. Procédé d'apprentissage selon la revendication 5, dans lequel ledit processus d'apprentissage est effectué en utilisant différentes fonctions de transformation h_{ou} h_{off} par rapport à un modèle ON donné dont la sortie cible est 1 et un modèle OFF donné dont la sortie cible est 0, en rendant maximal une somme des fonctions de transformation par rapport à tous les modèles, et en transformant ledit modèle d'entrée/sortie en un modèle linéairement séparable sur la base d'une équation de condition d'optimalisation pour rendre maximale la somme des fonctions de transformation, si ledit modèle d'entrée/sortie est linéairement inséparable.

13. Procédé d'apprentissage selon la revendication 5, comportant de plus les étapes consistant à :

déterminer un modèle d'apprentissage conformément à un processus prédéterminé à partir d'un modèle d'entrée/sortie d'origine et un modèle d'entrée/sortie réalisé par les neurones cachés qui ont subi un apprentissage, et
disposer le modèle d'apprentissage dans des neurones cachés qui n'ont pas subi d'apprentissage.

14. Procédé d'apprentissage selon la revendication 5, dans lequel ledit modèle est transformé en transformant un modèle OFF dont la sortie cible est 0 en un modèle ON dont la sortie cible est 1, et en modifiant le modèle OFF en un modèle ON, lorsque tous les éléments du modèle OFF ont une valeur égale ou supérieure à la valeur d'éléments correspondants d'un modèle ON.

15. Procédé d'apprentissage selon la revendication 5, dans lequel ledit ensemble de modèles est extrait en contrôlant un changement des coefficients d'association entre lesdites couches cachées et ladite couche d'entrée, et en extrayant un ensemble de modèles dans lequel les coefficients d'association ne changent pas et la sortie cible ne correspond pas avec la sortie réelle.

16. Procédé d'apprentissage selon la revendication 8, dans lequel ladite condition de finition est basée sur une valeur d'une fonction objet prédéterminée ayant un argument comportant lesdits coefficients d'association, ou un hypogradient de ladite fonction objet.

17. Procédé d'apprentissage selon la revendication 5, dans lequel lesdits coefficients d'association entre ladite couche de sortie et lesdits neurones cachés générés sont déterminés de manière à avoir un signe alterné dans l'ordre de la génération desdits neurones cachés générés.

FIG. 1

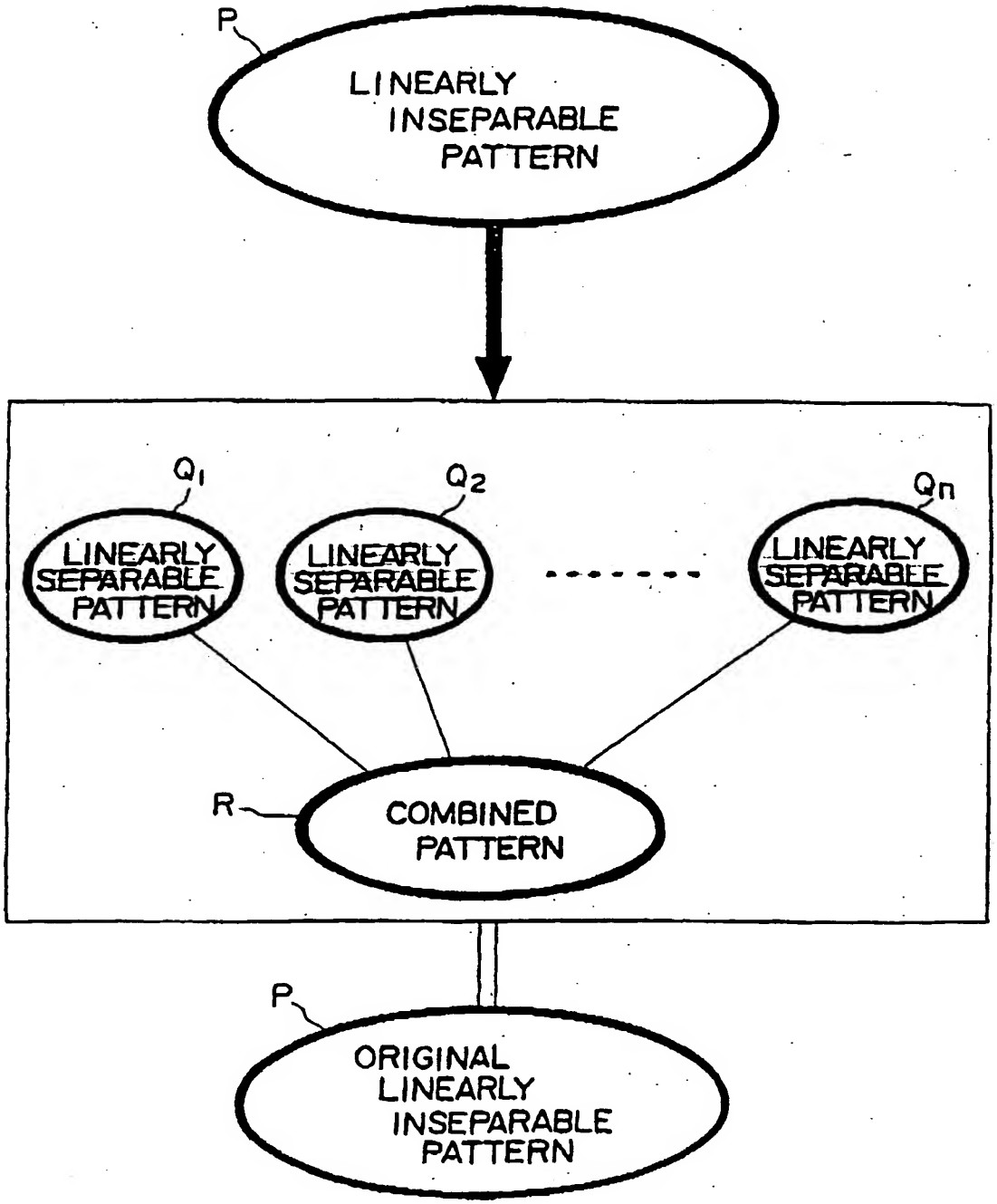

P
LINEARLY INSEPARABLE PATTERN
Q1
Q2
Qn
LINEARLY SEPARABLE PATTERN
LINEARLY SEPARABLE PATTERN
LINEARLY SEPARABLE PATTERN
R
COMBINED PATTERN
P
ORIGINAL LINEARLY INSEPARABLE PATTERN

FIG. 2

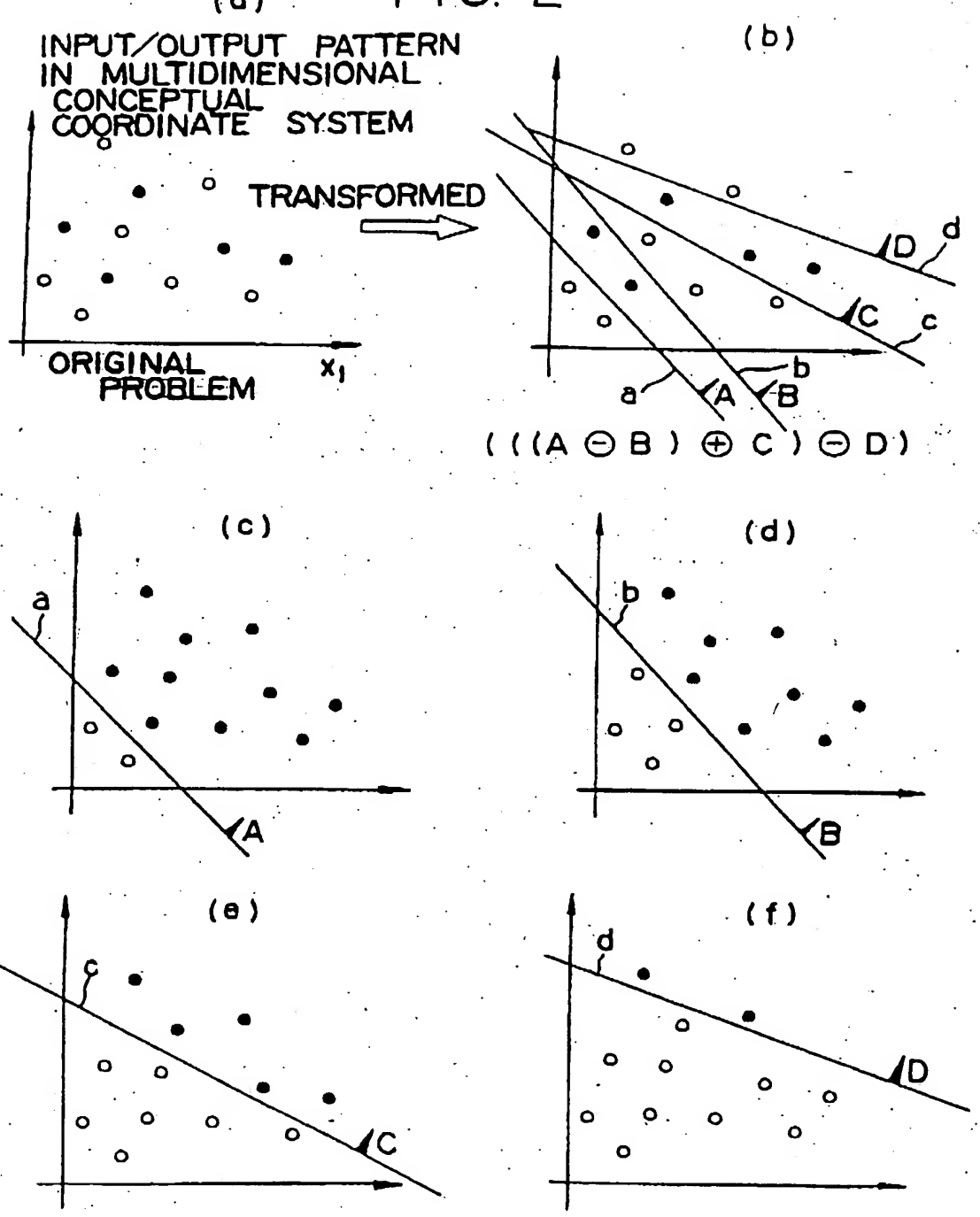

(a)
INPUT/OUTPUT PATTERN IN MULTIDIMENSIONAL CONCEPTUAL COORDINATE SYSTEM
TRANSFORMED
ORIGINAL PROBLEM
x_1
(b)
D
d
C
c
a
A
b
B
(((A ⊖ B) ⊕ C) ⊖ D)
(c)
a
A
(d)
b
B
(e)
c
C
(f)
d
D

FIG. 3

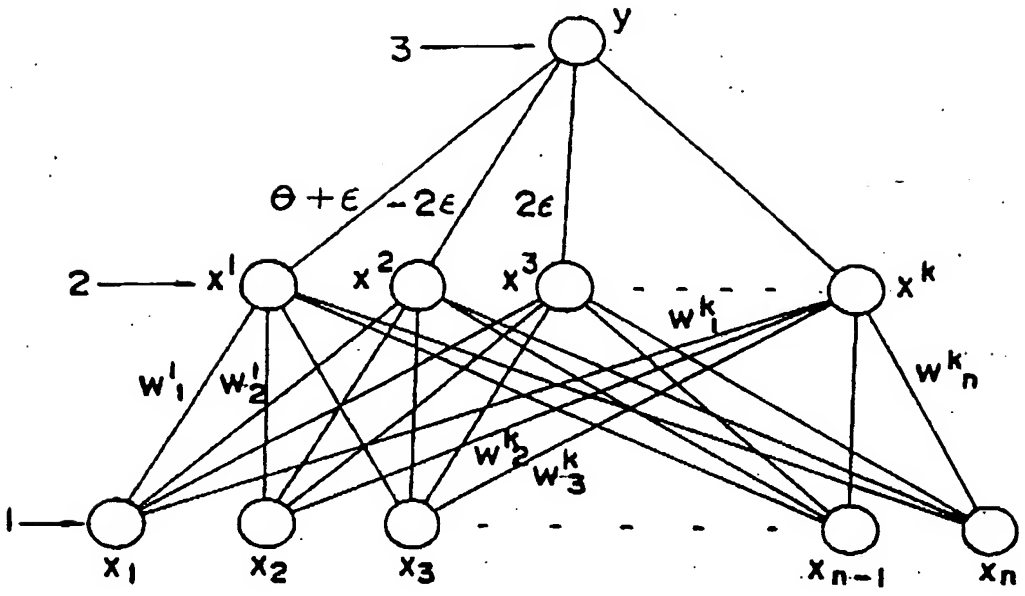

y
3
θ+ε
-2ε
2ε
2
x1
x2
x3
xk
wk1
wkn
w11
w12
wk2
wk3
1
x1
x2
x3
xn-1
xn

FIG. 4

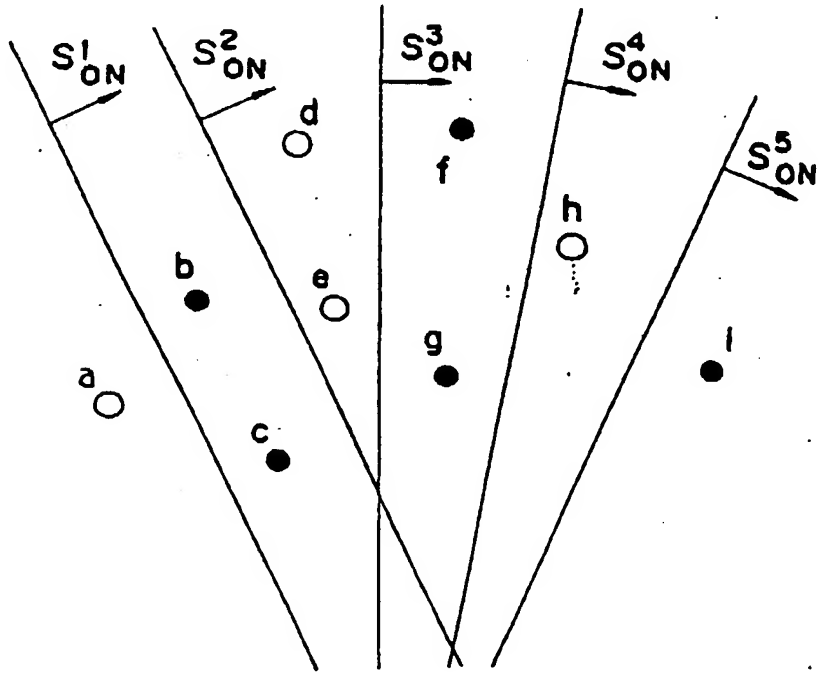

S1ON
S2ON
S3ON
S4ON
S5ON
a
b
c
d
e
f
g
h
i

FIG. 5

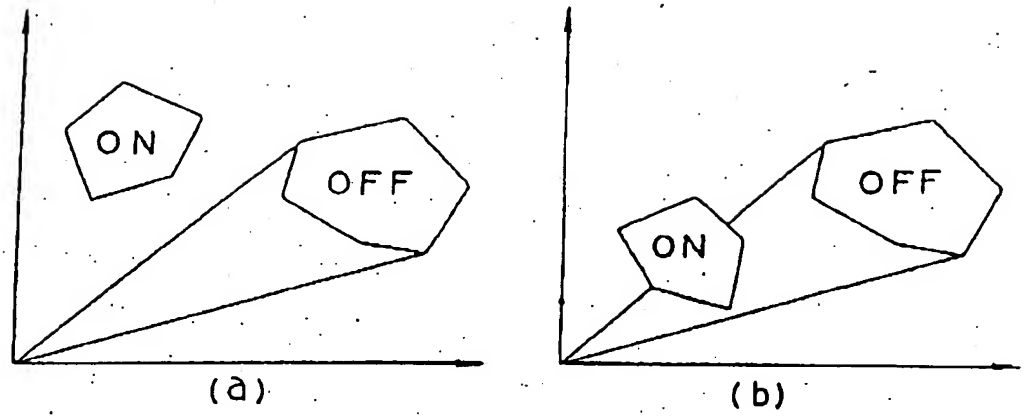

ON
OFF
(a)
ON
OFF
(b)

FIG. 6

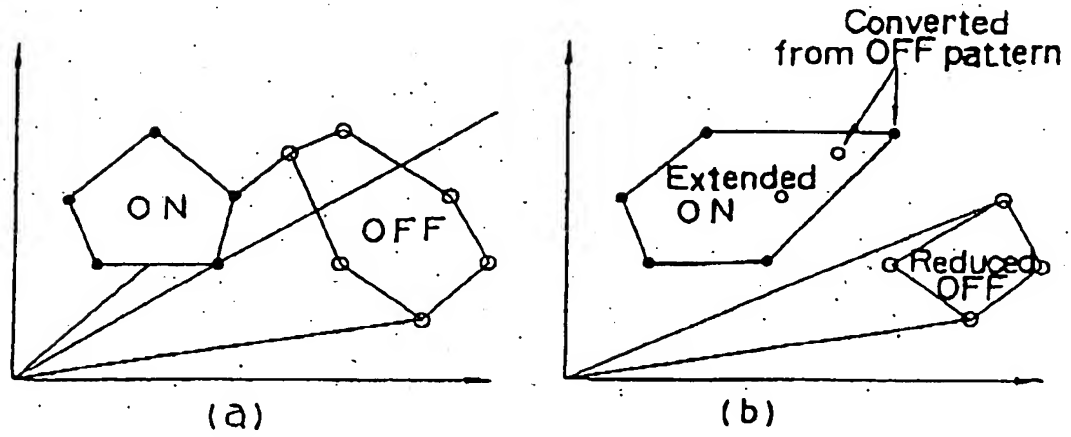

ON
OFF
(a)
Converted
from OFF pattern
Extended
ON
Reduced
OFF
(b)

FIG. 7

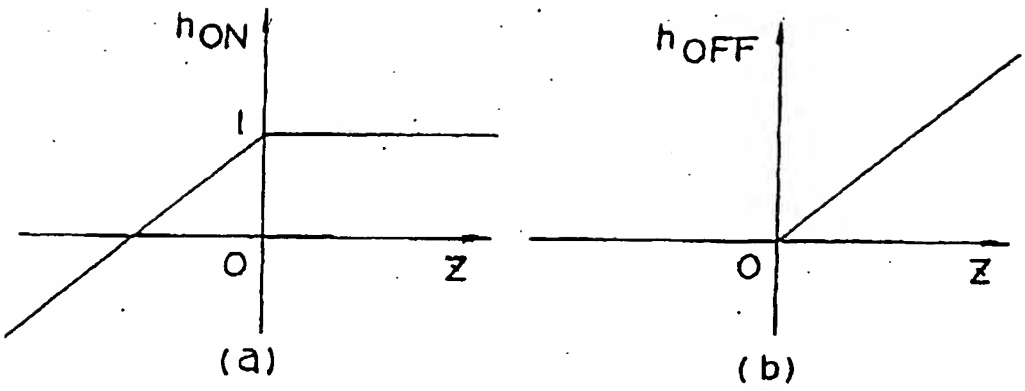


FIG. 8

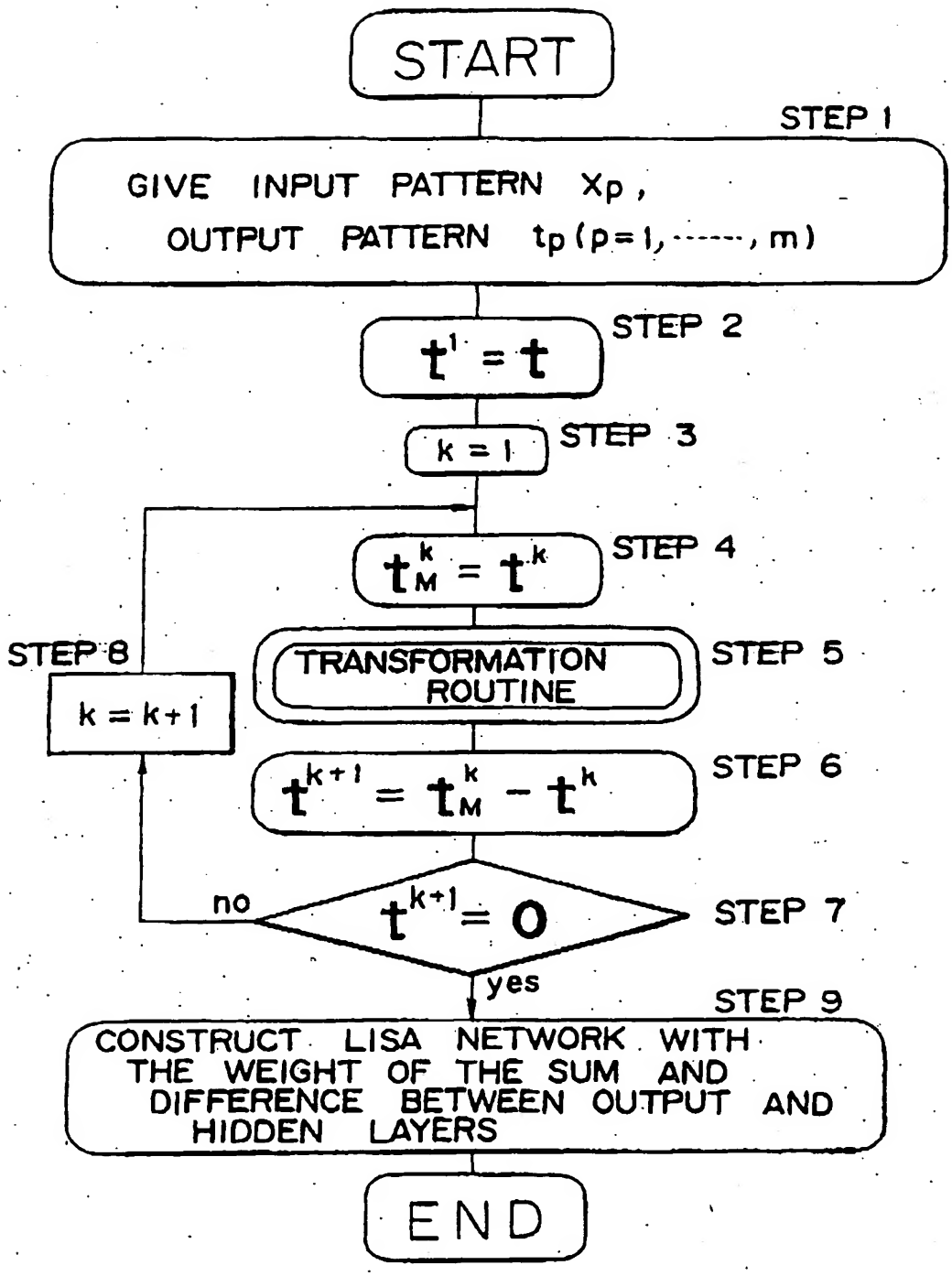

START
STEP 1
GIVE INPUT PATTERN Xp ,
OUTPUT PATTERN tp (p=1, ·····, m)
STEP 2
t¹ = t
STEP 3
k = 1
STEP 4
t_M^k = t^k
STEP 5
TRANSFORMATION ROUTINE
STEP 6
t^(k+1) = t_M^k - t^k
STEP 7
t^(k+1) = 0
no
yes
STEP 8
k = k+1
STEP 9
CONSTRUCT LISA NETWORK WITH THE WEIGHT OF THE SUM AND DIFFERENCE BETWEEN OUTPUT AND HIDDEN LAYERS
END

FIG. 9

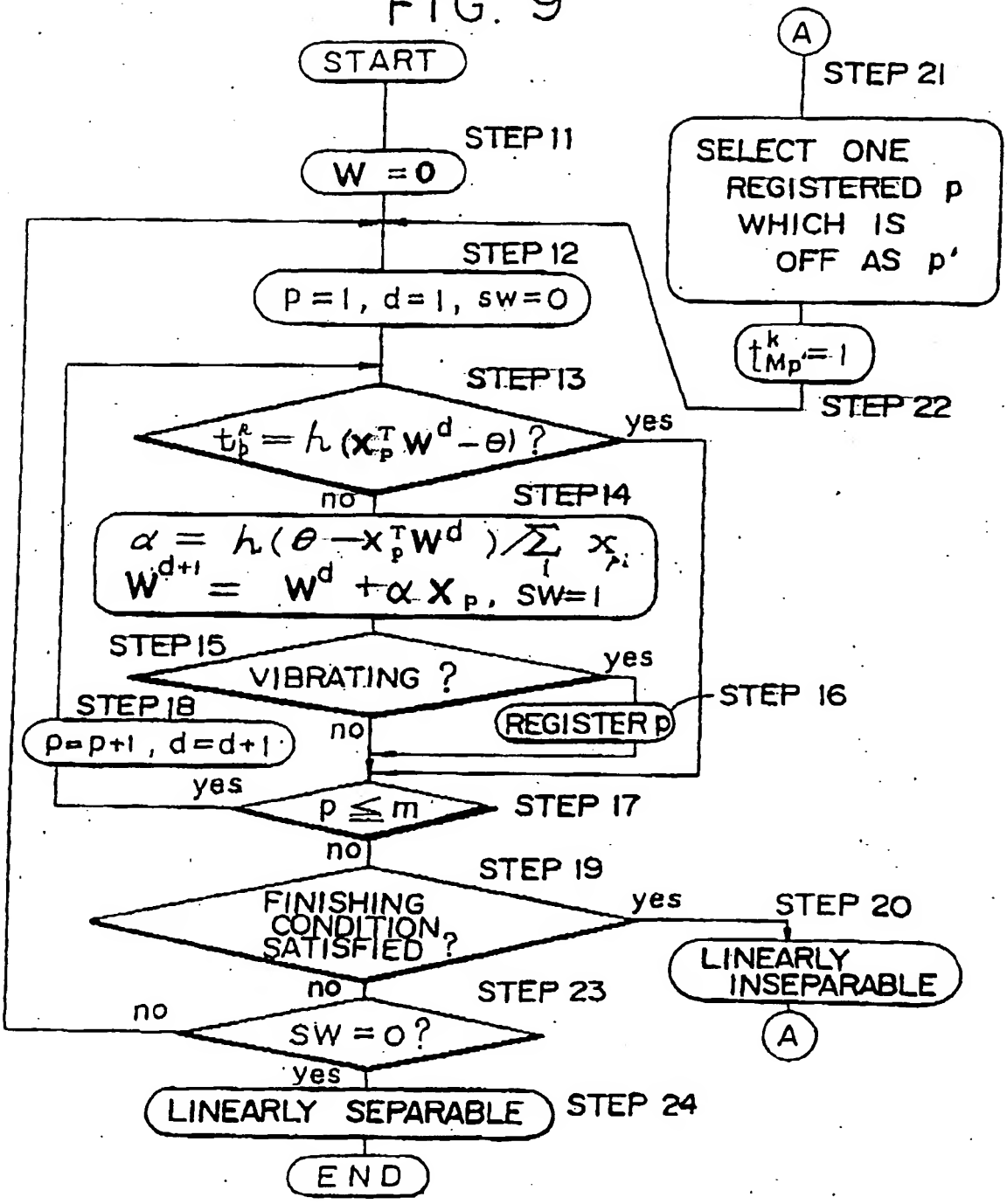
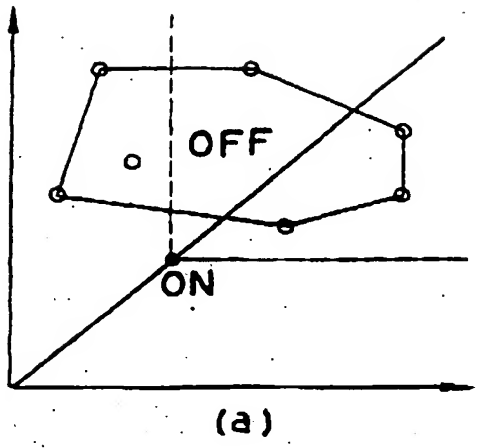

START
STEP 11
W = 0
STEP 12
p = 1, d = 1, sw = 0
STEP 13
$t_p^k = h(x_p^T w^d - \theta)$?
yes
no
STEP 14
$\alpha = h(\theta - x_p^T w^d) / \sum_i x_{pi}$
$w^{d+1} = w^d + \alpha x_p$, SW = 1
STEP 15
VIBRATING ?
yes
STEP 16
REGISTER p
STEP 18
p = p+1, d = d+1
no
yes
$p \leq m$
STEP 17
no
STEP 19
FINISHING CONDITION SATISFIED ?
yes
STEP 20
LINEARLY INSEPARABLE
A
no
STEP 23
SW = 0 ?
no
yes
LINEARLY SEPARABLE
STEP 24
END
A
STEP 21
SELECT ONE REGISTERED p WHICH IS OFF AS p'
$t_{Mp'}^k = 1$
STEP 22

FIG. 10



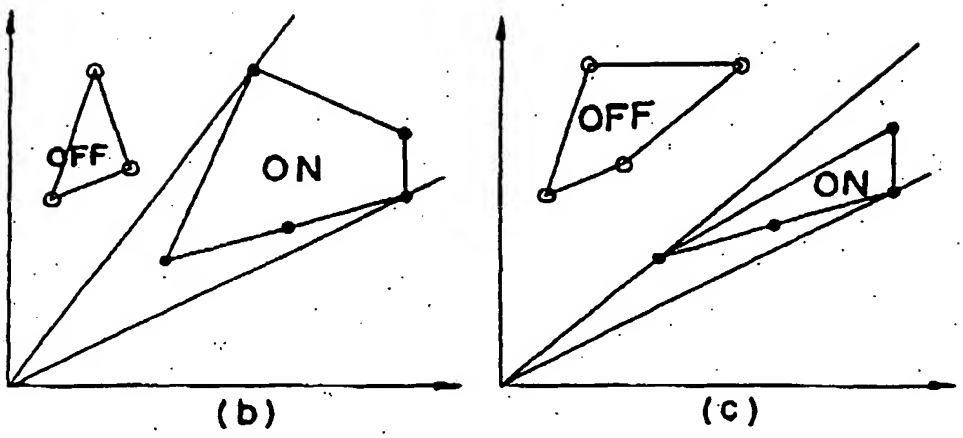
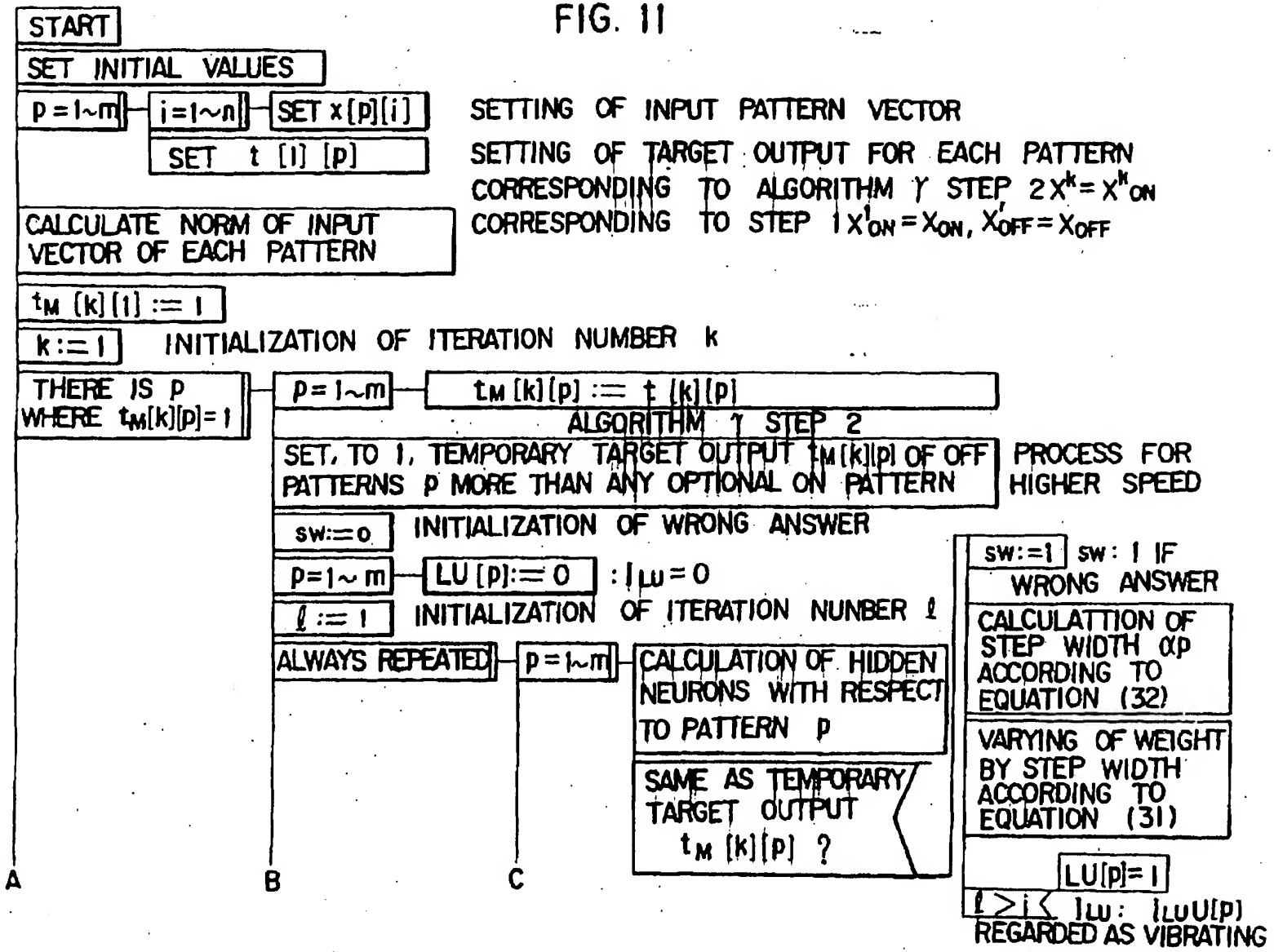
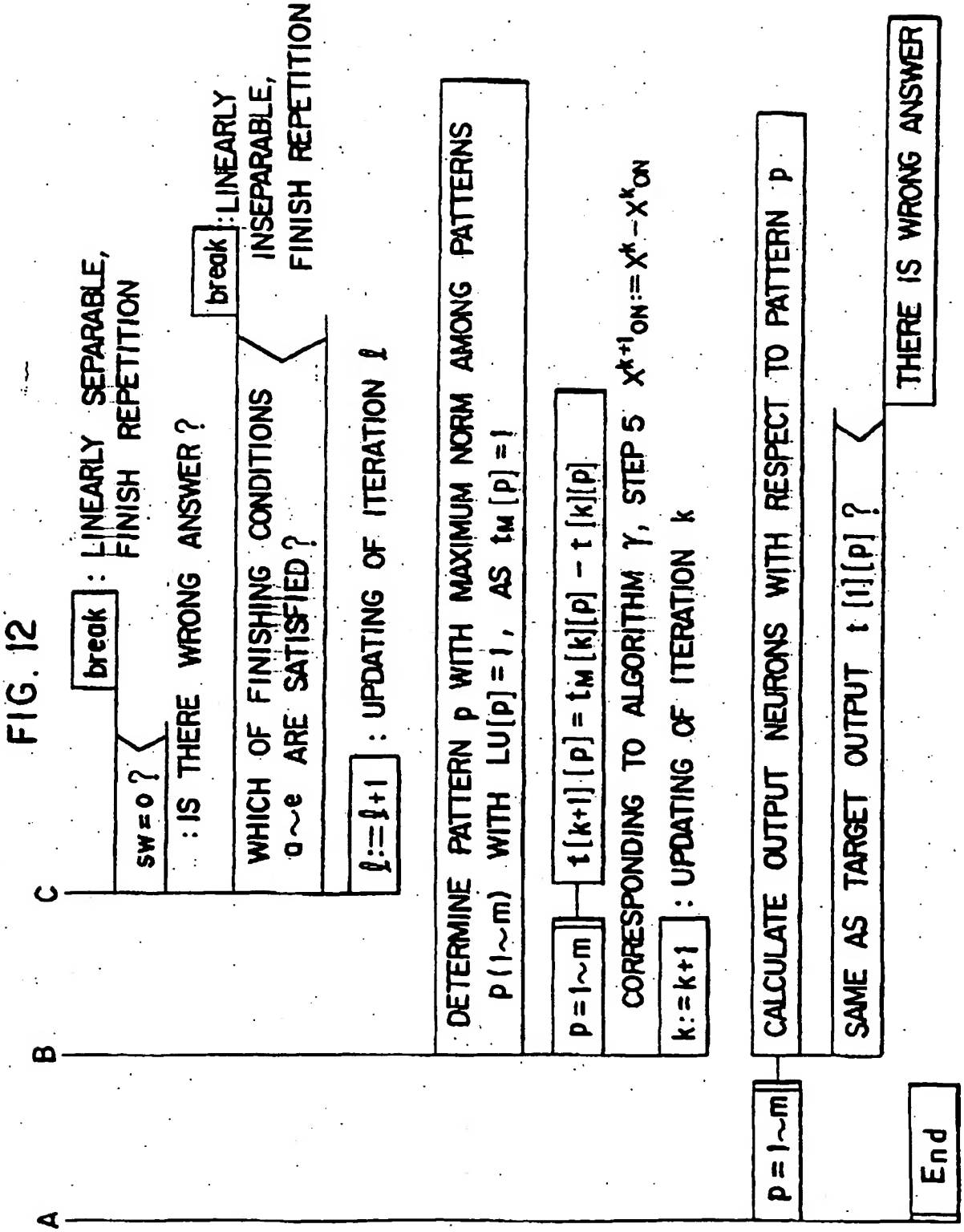


FIG. 11



EP 0 521 729 B1

FIG. 12


A
B
C
End
p=1~m
CALCULATE OUTPUT NEURONS WITH RESPECT TO PATTERN p
SAME AS TARGET OUTPUT t[l][p] ?
THERE IS WRONG ANSWER
k:=k+1 : UPDATING OF ITERATION k
CORRESPONDING TO ALGORITHM γ, STEP 5 $X^{k+1}_{ON} := X^k - X^k_{ON}$
p=1~m
t[k+1][p] = t_M[k][p] - t[k][p]
DETERMINE PATTERN p WITH MAXIMUM NORM AMONG PATTERNS p(1~m) WITH LU[p]=1, AS t_M[p]=1
ℓ:=ℓ+1 : UPDATING OF ITERATION ℓ
WHICH OF FINISHING CONDITIONS a~e ARE SATISFIED ?
break : LINEARLY INSEPARABLE, FINISH REPETITION
: IS THERE WRONG ANSWER ?
sw=0 ?
break : LINEARLY SEPARABLE, FINISH REPETITION

DESCRIPTION OF VARIABLES FIG. 13

k	ITERATION VARIABLE OF ALGORITHM γ (PATTERN ASSIGNING ALGORITHM FOR HIDDEN LAYER NEURONS)
ℓ	ITERATION VARIABLE FOR ALGORITHM δ^k (LEARNING ALGORITHM FOR HIDEN LAYER NEURONS)
p	VARIABLE REPRESENTING PATTERN SUFFIX
i	VARIABLE REPRESENTING SUFFIX TO ELEMENTS OF INPUT VECTOR OF LEARNING PATTERN
sw	VARIARBLE REPRESENTING WRONG ANSWER
t [k][p]	TARGET OUTPUT FOR kTH NEURON OF HIDDEN LAYER. CORRESPONDS TO X^k_{ON}, X^k_{OFF} OF ALGORITHM γ. PATTERN p WITH t[k][p] BEING 1 IS CONTAINED IN X^k_{ON}, AND PATTERN p WITH t[k][p] BEING 0 IS CONTAINED IN X^k_{OFF}.
t_M[k][p]	LINEARLY SEPARABLE LEARNING PATTERN WITH RESPECT TO kTH NEURON OF HIDDEN LAYER GENERATED BY ALGORITHM. CORRESPONDS TO X^k OF ALGORITHM γ AND Y^k_{ON}, Y^k_{OFF} OF ALGORITHM δ^k. PATTERN p WITH t_M[k][p] BEING 1 IS CONTAINED IN Y^k_{ON}, X^k, AND PATTERN p WITH t_M[k][p] BEING 0 IS CONTAINED IN Y^k_{OFF}.
LU [p]	CORRESPONDS TO I_{LU} OF ALGORITHM δ^k. PATTERN p WITH LU[p] BEING 1 IS CONTAINED IN I_{LU}

SYMBOLS USED ARE DFFINED AS FOLLOWS

- INDICATES REPETITION.
- INDICATES BRANCHES. IF TRUE, PROCESSING GOES TO UPPER STEP, AND IF FALSE, PROCESSING GOES TO LOWER STEP. WHEN FINISHED, PROCESSING PROCEEDS TO NEXT STEP.
- break — INDICATES CANCELLING OF PRESENTLY EFFECTED REPETITION, AND PROCESSING OF NEXT STEP
- i=1~n — tM[i][i] = t[k][i] — INDICATES VARYING OF i FROM 1 TO n, AND SUBSTITUTION OF t[k][i] INTO t_M[i][i].

"=" INDICATES COMPARISON.
":=" INDICATES SUBSTITUTION.

EP 0 521 729 B1

FIG. 14

FLOWCHART OF ALGORITHM γ

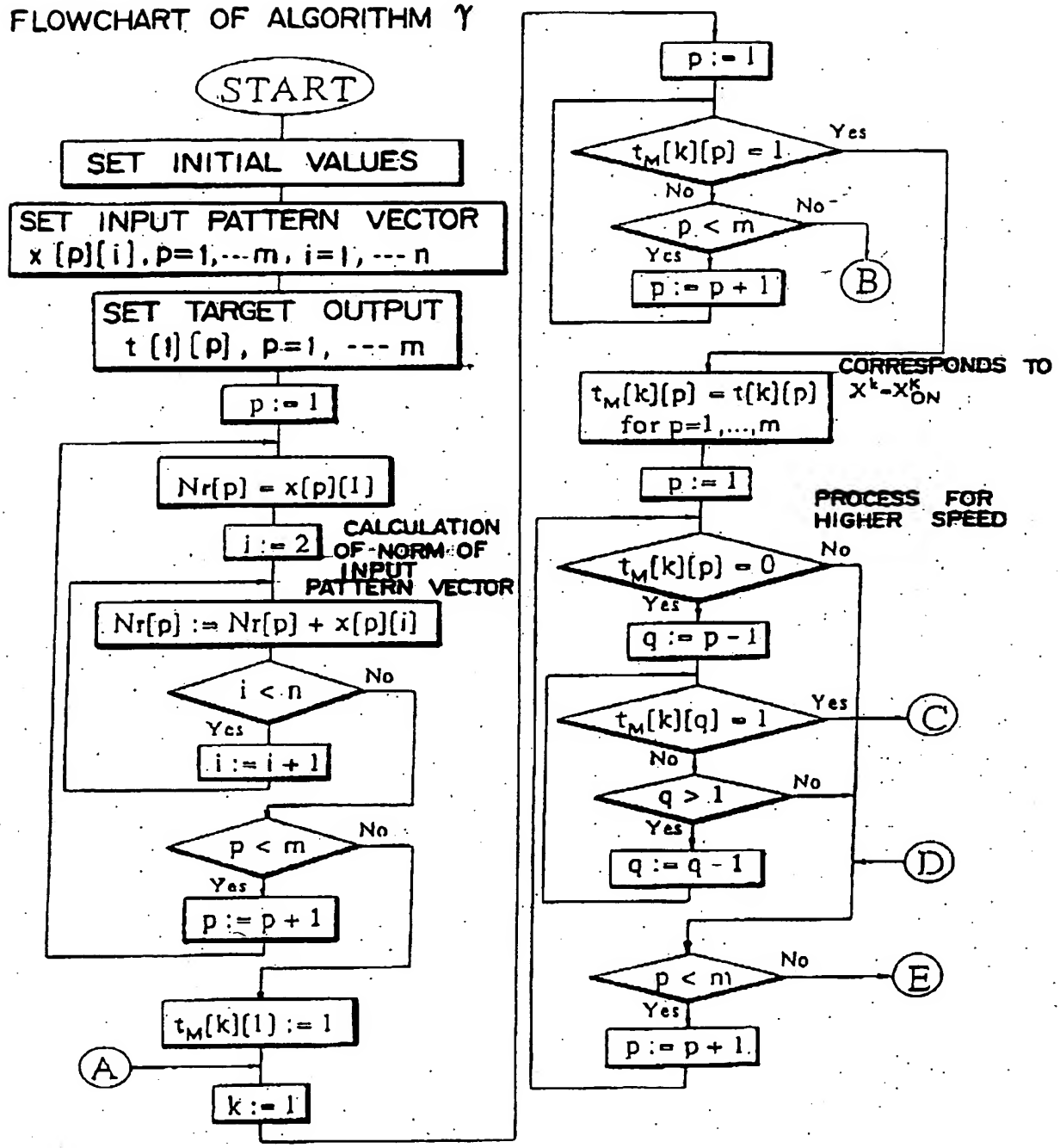


FIG. 15

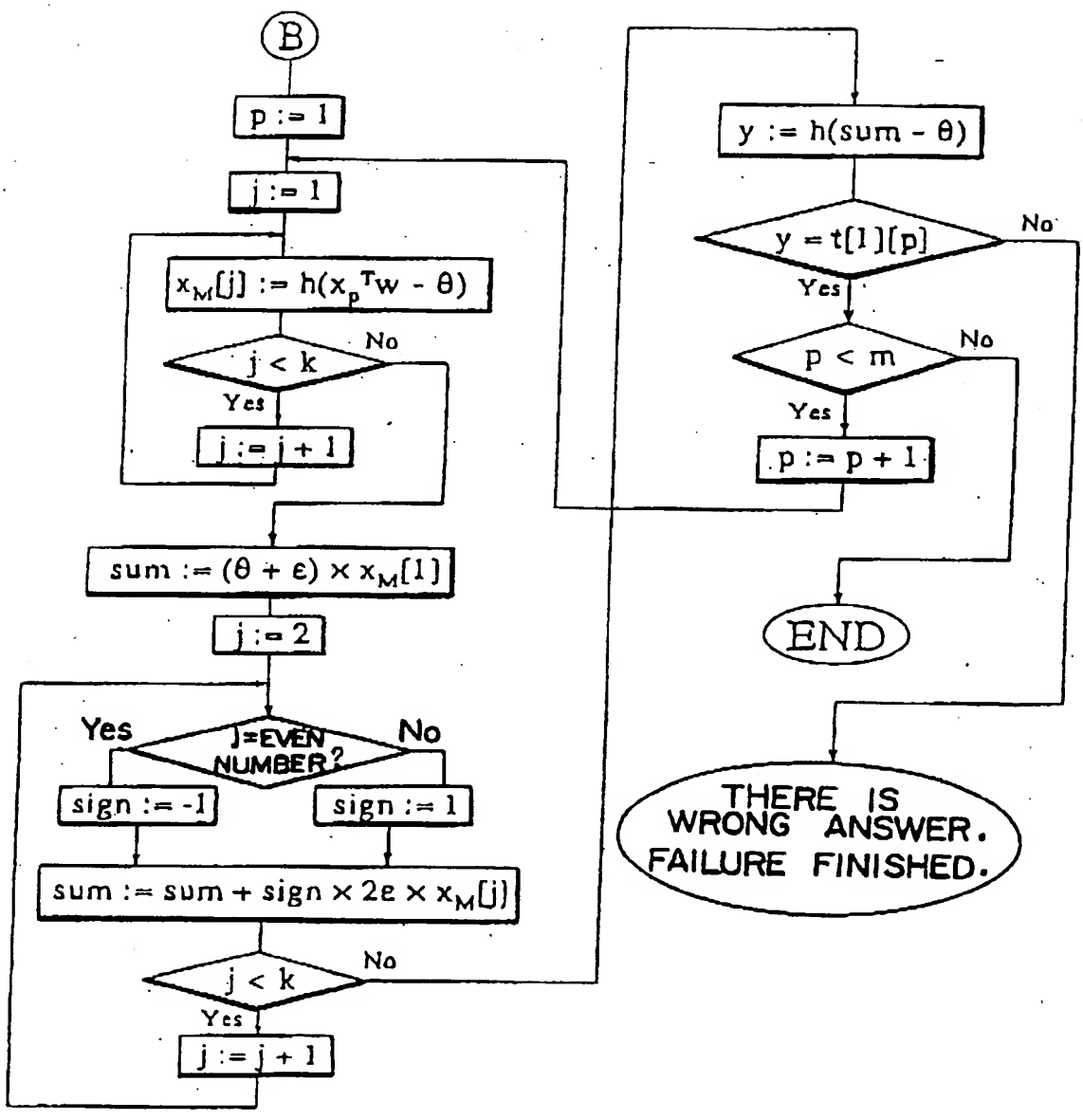

B
p := 1
j := 1
$x_M[j] := h(x_p^T w - \theta)$
j < k
No
Yes
j := j + 1
sum := (θ + ε) × $x_M[1]$
j := 2
Yes
j=EVEN NUMBER?
No
sign := -1
sign := 1
sum := sum + sign × 2ε × $x_M[j]$
j < k
No
Yes
j := j + 1
y := h(sum - θ)
y = t[1][p]
No
Yes
p < m
No
Yes
p := p + 1
END
THERE IS WRONG ANSWER. FAILURE FINISHED.

FIG. 16

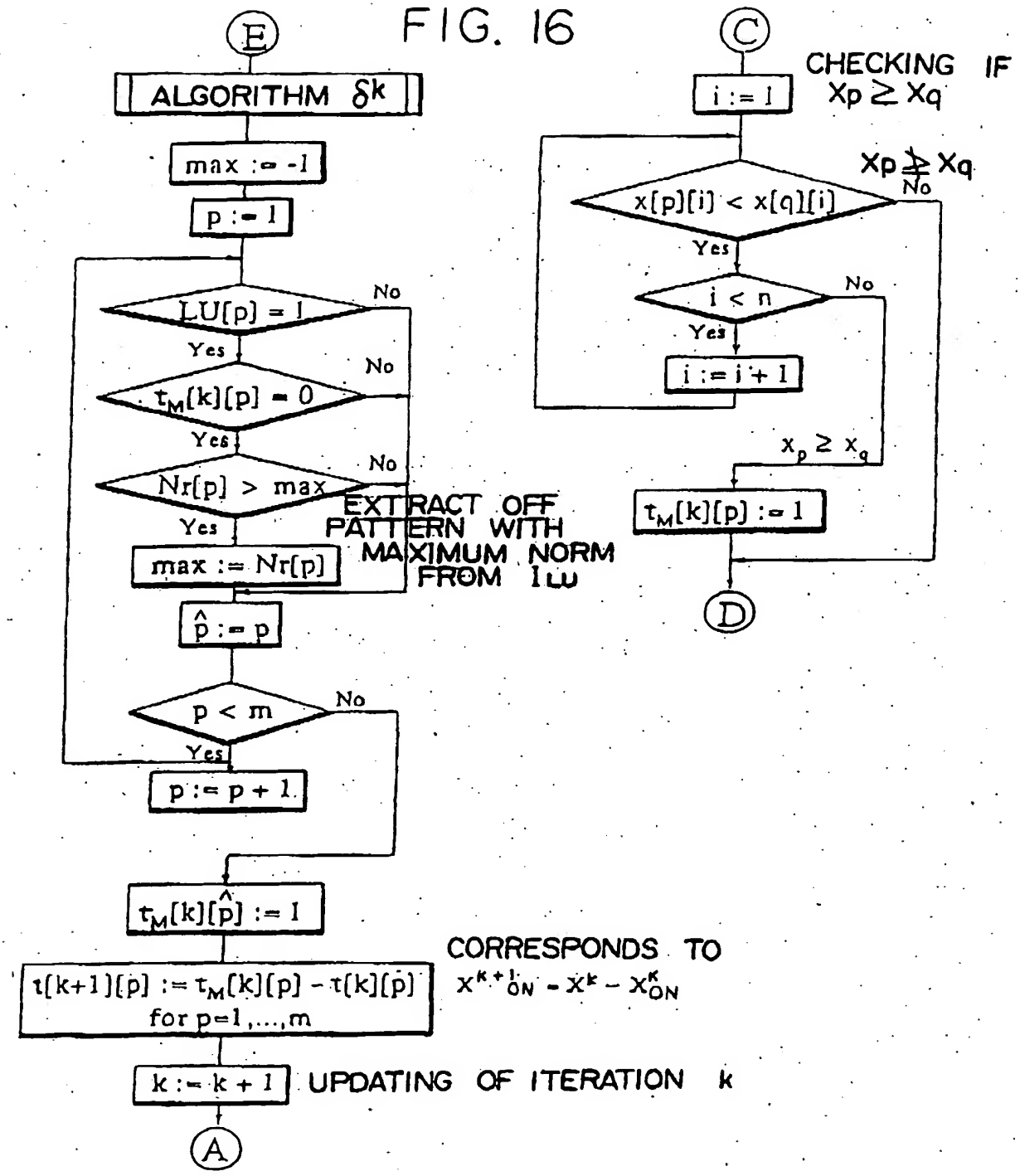

E
ALGORITHM δ^k
max := -1
p := 1
LU[p] = 1
No
Yes
$t_M[k][p] = 0$
No
Yes
Nr[p] > max
No
Yes
max := Nr[p]
$\hat{p} := p$
p < m
No
Yes
p := p + 1
$t_M[k][\hat{p}] := 1$
$\tau[k+1][p] := t_M[k][p] - \tau[k][p]$ for p=1,...,m
k := k + 1
A
EXTRACT OFF PATTERN WITH MAXIMUM NORM FROM Iω
C
i := 1
CHECKING IF $Xp \geq Xq$
$Xp \not\geq Xq$
No
x[p][i] < x[q][i]
Yes
i < n
No
Yes
i := i + 1
$x_p \geq x_q$
$t_M[k][p] := 1$
D
CORRESPONDS TO $X^{k+1}_{ON} - X^k - X^k_{ON}$
UPDATING OF ITERATION k

FIG. 17

FLOWCHART OF ALGORITHM δk

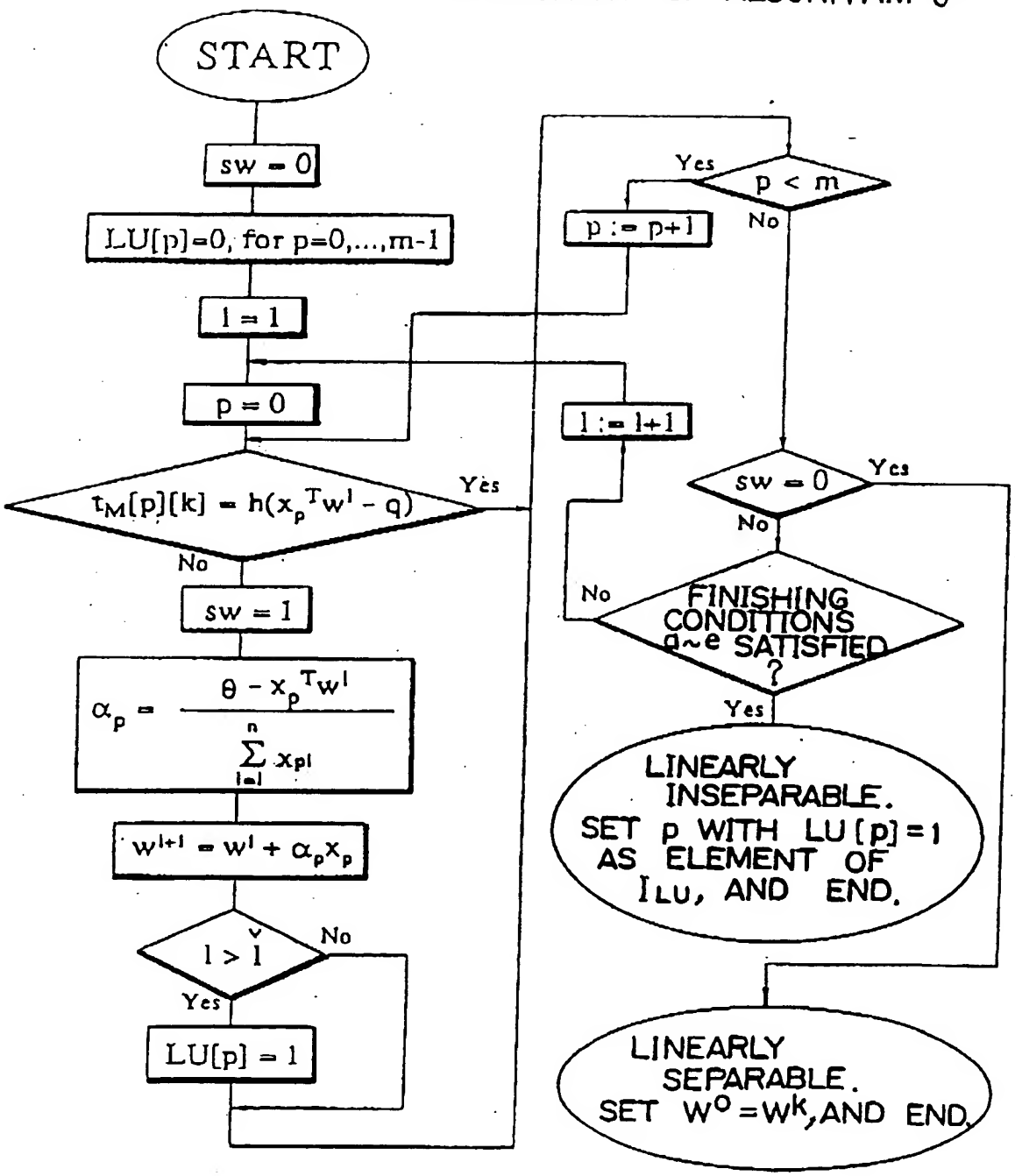


FIG. 18

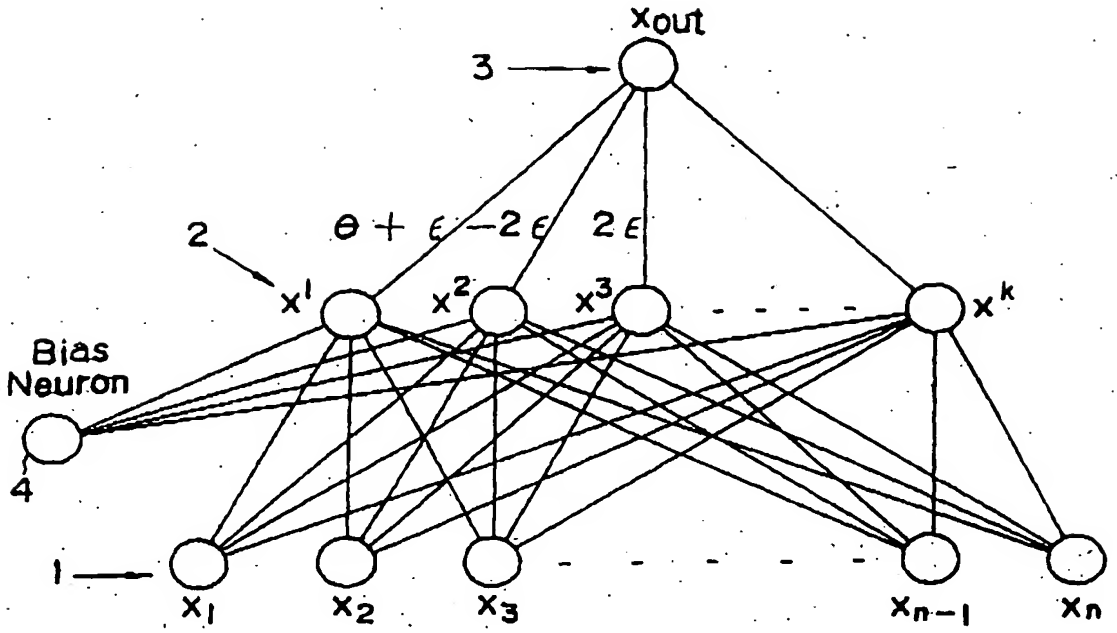

Xout
3
2
θ + ε
−2ε
2ε
x1
x2
x3
xk
Bias Neuron
4
1
x1
x2
x3
xn−1
xn

FIG. 19

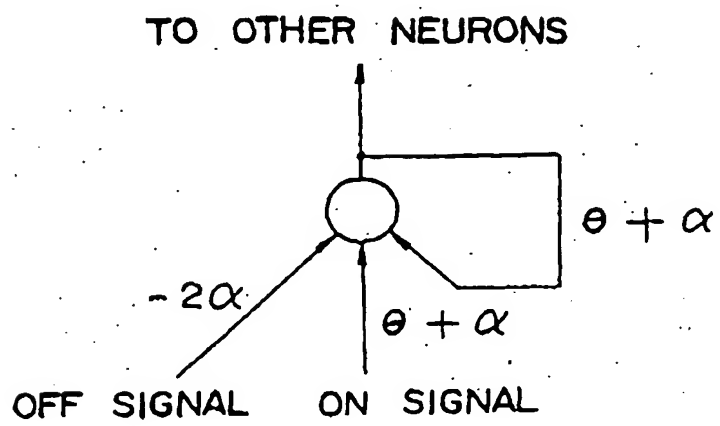

TO OTHER NEURONS
θ + α
-2α
θ + α
OFF SIGNAL
ON SIGNAL

FIG. 20

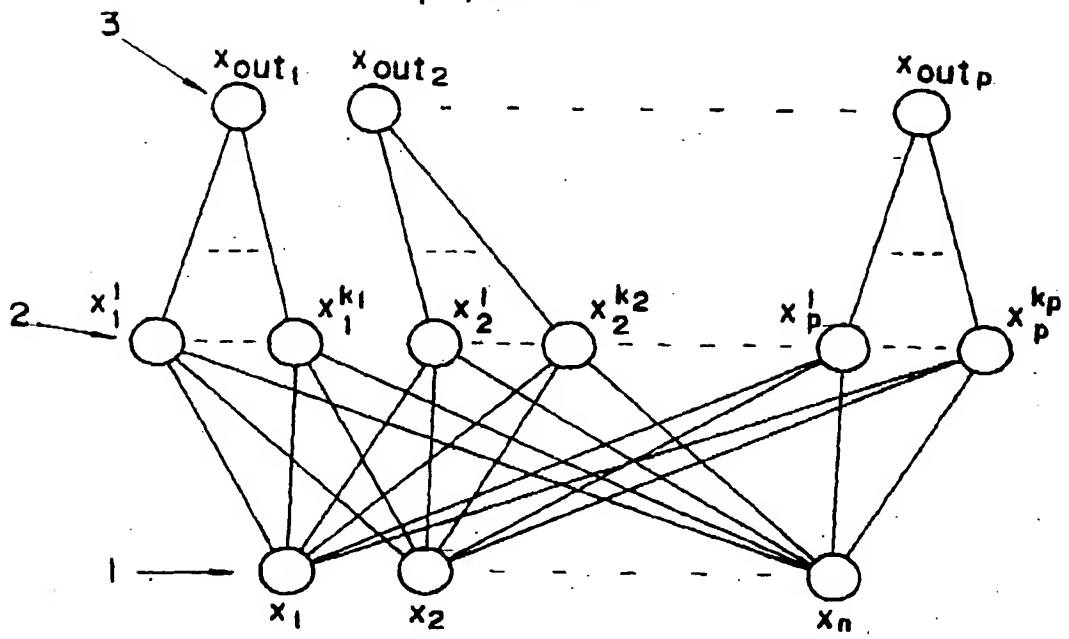

3
x_{out_1}
x_{out_2}
x_{out_p}
2
x_1^1
$x_1^{k_1}$
x_2^1
$x_2^{k_2}$
x_p^1
$x_p^{k_p}$
1
x_1
x_2
x_n

FIG. 21

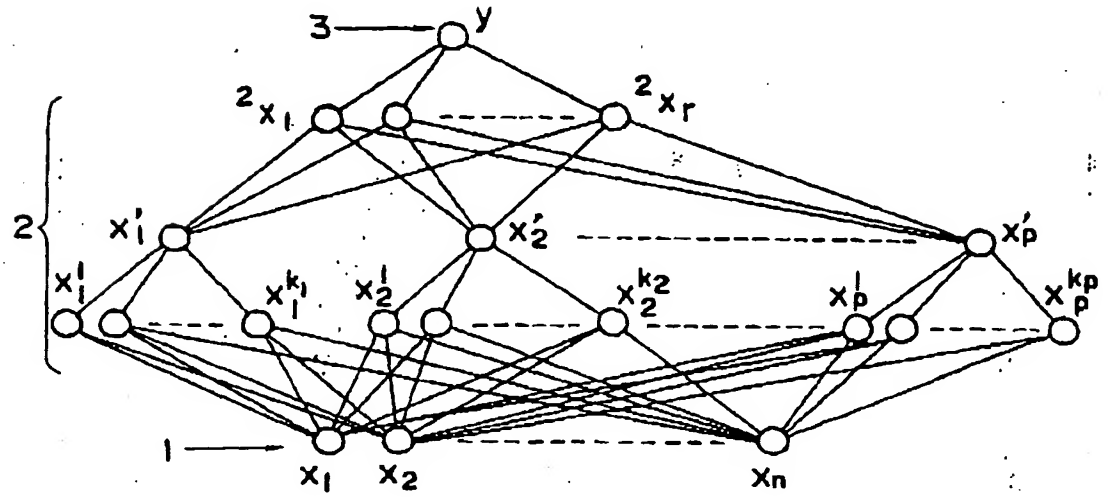

3
y
2
$^2x_1$
^2x_r
x_1'
x_2'
x_p'
x_1^1
$x_1^{k_1}$
x_2^1
$x_2^{k_2}$
x_p^1
$x_p^{k_p}$
1
x_1
x_2
x_n

FIG. 22

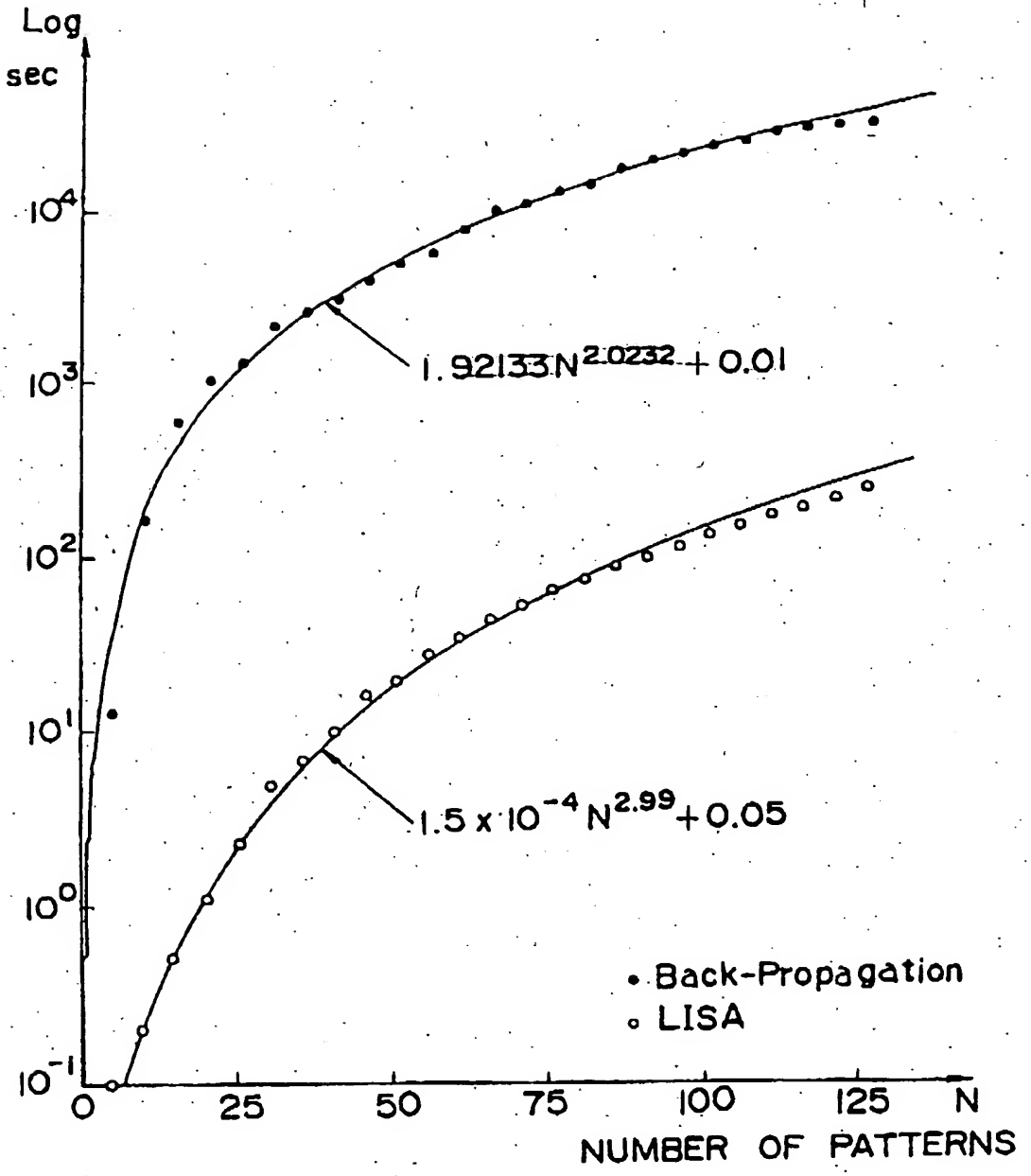
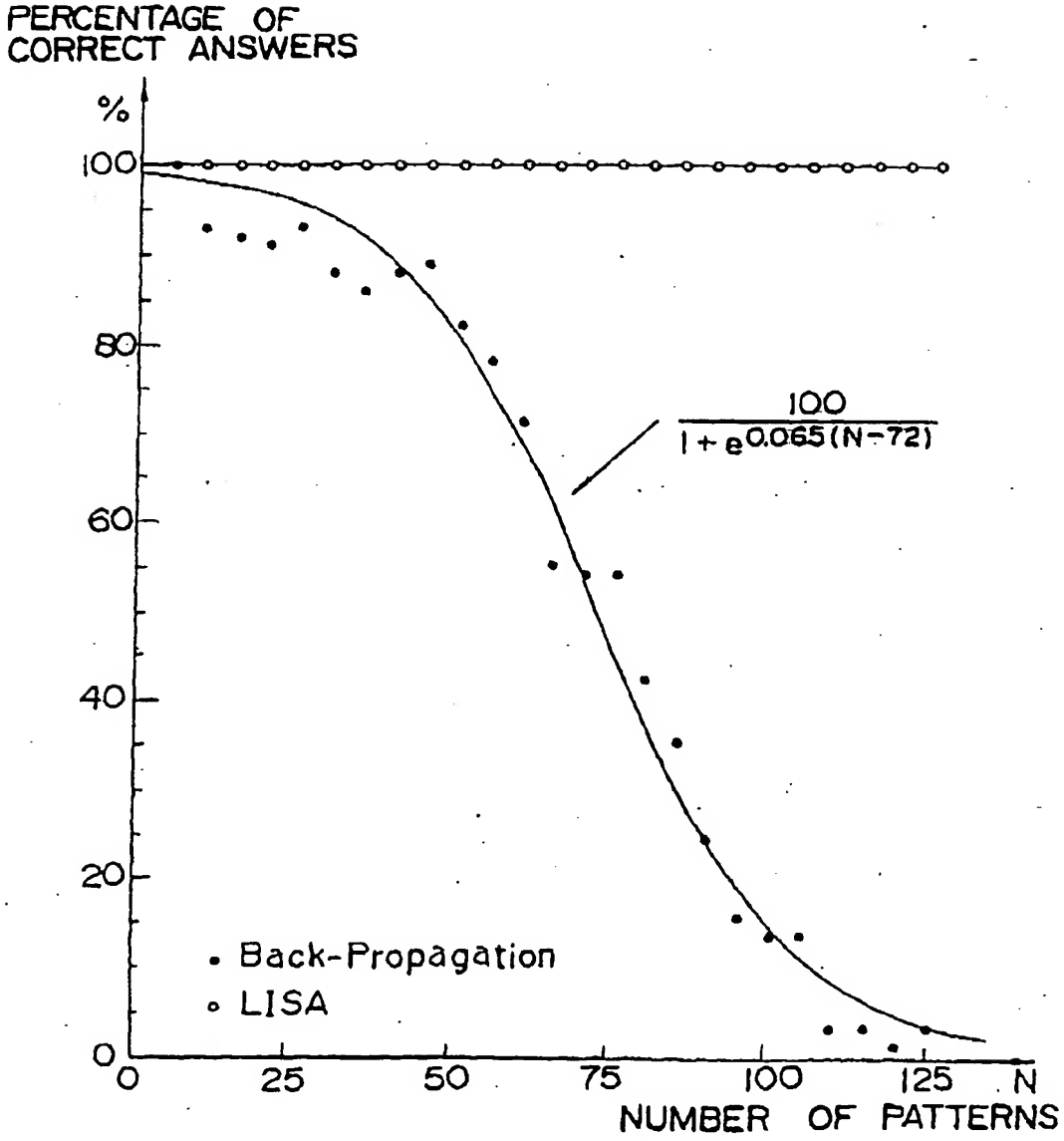

Log
sec
10^4
10^3
10^2
10^1
10^0
10^{-1}
$1.92133N^{2.0232}+0.01$
$1.5 \times 10^{-4} N^{2.99}+0.05$
• Back-Propagation
◦ LISA
0
25
50
75
100
125
N
NUMBER OF PATTERNS

FIG. 23


PERCENTAGE OF CORRECT ANSWERS
%
100
80
60
40
20
0
0
25
50
75
100
125
N
NUMBER OF PATTERNS
100 / 1 + e^0.065(N-72)
• Back-Propagation
◦ LISA